

[illegible]

QA 278.3 .M3 1990
McArdle, Jack J.
RAMpath

**Wilkins Biomedical Library
Naval Health Research Center
San Diego, CA**

Data Transforms' Software is copyrighted and all rights are reserved by Data Transforms, Inc. The distribution and sale of this product are intended for use by the original purchaser at a single computer station only. (Multi-station use requires a separate license.)

Lawful users of this program are hereby licensed only to read the program from its medium into memory of a computer solely for the purpose of executing the program.

Copies of the software may be made for archival or backup purposes only. The software may be transferred to a single hard disk only, provided the original be kept for archival or backup purposes only.

The software may not be copied, modified, altered or changed, in whole or in part other than as provided by and through the normal use of the software and the restrictions detailed in this license.

LIMITED WARRANTY

Data Transforms' Software is sold on an "as is" basis without any warranty either express, statutory, or implied as to performance, quality, merchantability, or fitness for any particular purpose. The entire risk is with the buyer. Should the software or manual prove defective following purchase, the buyer (not Data Transforms, Inc., its retailer or its distributor) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages resulting from any defect in the software or manual, even if Data Transforms, Inc. has been informed of such possibility of damages. In no event shall Data Transforms, Inc. or anyone else who has been involved in the creation or production of Data Transforms software be liable for indirect, special or consequential damages (including loss of profit) resulting from the use of Data Transforms' software. Some states do not allow the exclusion or limitation of incidental or consequential damages so the above limitation or exclusion may not apply to you.

RAMpath (c) Copyright 1986-1990 Steven M. Boker and J. J. McArdle

DATA TRANSFORMS, INC.

616 Washington St.

Denver, CO 80203

303-832-1501

rampath@datran2.uunet.uu.net

Printed in U.S.A.

RAMpathtm is a trademark of Data Transforms, Inc.

Printrixtm is a trademark of Data Transforms, Inc.

SAS/CALISTtm is a trademark of SAS Institute.

LISRELtm is a trademark of International Educational Services.

RAMpath Path Diagram Software

(c) 1986-1990 by J.J. McArdle & Steven M. Boker
All Rights Reserved

Table of Contents

Part I: An Introduction to RAMpath

1: The RAMpath Program	I-2
2: Running RAMpath Quickly	I-3
3: Using this Manual	I-4
4: Our Historical Development	I-5
5: Some Acknowledgements	I-6

Part P: Programming with RAMpath

1: A simple example of RAMpath	P-2
2: RAMpath program language	P-4
3: Drawing path diagrams	P-6
4: Tracing the rules of path analysis	P-11
5: Matrix input and output	P-15
6: Additional examples on disk	P-19

Part M: Structural Modeling with RAM Diagrams

1: Introduction to RAM structural modeling	M-2
2: Structural equation models and path diagrams	M-3
3: The algebraic principles of RAM theory	M-8
4: The graphic principles of RAM theory	M-12
5: RAMpath versus traditional path diagrams	M-15
6: Statistical principles	M-17
7: Drawing path diagrams from SAS and LISREL	M-19

Part D: The Dictionary of RAMpath Commands

1: Introduction	D-2
2: Running RAMpath	D-2
3: RAMpath Language Rules	D-3
4: Adding Comments	D-4
5: Aliasing Words for Numbers	D-5
6: DRAW_BRIDGES Command	D-6
7: DRAW_MODEL Command	D-7
8: INPUT_CLEAR Command	D-8
9: INPUT_MODEL Command	D-9
10: INPUT_MODEL_MATRIX Command	D-14
11: OUTPUT_BRIDGES Command	D-16
12: OUTPUT_MODEL Command	D-17
13: OUTPUT_MODEL_MATRIX Command	D-18
14: SAVE_MODEL Command	D-19
15: SET_COLOR Command	D-20
16: SET_EQUAL Command	D-21
17: SET_ERROR Command	D-22
18: SET_LABEL Command	D-23
19: SET_PRINT Command	D-24
20: SET_SHAPE Command	D-25
21: SET_SLING_ROTATE Command	D-26
22: SET_TITLE Command	D-27
23: SET_VALUE Command	D-28
24: SET_WIDTH Command	D-29
25: #DEFINE Command	D-30

Part R: Selected RAMpath References R-1

Quick Reference Card

A Summary of all RAMpath commands
The nine algebraic principles of RAM theory
The six graphic principles of RAM theory

List of RAMpath Tables

[M.3.1]:	RAM algebraic notation (from McArdle & McDonald, 1984)	M-9
[M.3.2]:	The latent variable path model using RAM algebra (see Figure [P.3.1])	P-18
[M.4.1]:	The Graphic Specification Rules for Linear Structural Equation Models	M-12
[M.4.2]:	The Graphic Action Rules for Non-linear Structural Expectations	M-13
[M.5.1]:	Some differences between RAM notation and Traditional Path Diagrams	M-16

List of RAMpath Files

[P.1.1]:	A simple input file for RAMpath _____	P-2
[P.1.2]:	An example of an output textfile generated by RAMpath _____	P-3
[P.3.1]:	The RAMpath input script for the latent variable path model _____	P-8
[P.3.2]:	A RAMpath output text file for the latent variable path model _____	P-10
[P.4.1]:	The text output produced by OUTPUT_BRIDGES _____	P-12
[P.4.2]:	Text output for the OUTPUT_BRIDGES of the Variance term _____	P-14
[P.5.1]:	RAMpath matrix input of the latent variable path model _____	P-17
[P.5.2]:	RAMpath matrix output of the latent variable path model _____	P-18
[M.7.1]:	An example of a SAS DATA step creating a RAM file (SAS2RAM.SAS) _____	M-20
[M.7.2]:	The resulting SAS output file (FROMSAS.RAM) _____	M-21
[M.7.3]:	LISREL-7 Input file with save options (LIS2SAV.L7) _____	M-22
[M.7.4]:	LISREL-7 Save file output (LIS2SAV.L7) and Revised LISREL output using RAMpath notation (LIS2RAM.RAM) _____	M-23

List of RAMpath Figures

[P.1.1]:	A very simple first example of RAMpath _____	P-3
[P.3.1]:	Prototypical Latent Variable Model _____	P-7
[P.4.1]:	Bridge #1 $K \rightarrow I \rightarrow 7 \rightarrow 4 = 0.210000$ _____	P-13
[M.2.1]:	Spearman's (1904) Single Common Factor Model _____	M-3
[M.2.2]:	Wright's (1921, 1934 p.177) First Path Analysis Model _____	M-4
[M.2.3]:	Bridge #1 $K \rightarrow 4 \rightarrow 6 \rightarrow 6 \rightarrow 5 \rightarrow 2 = 0.009500$ _____	M-5
[M.2.4]:	The Hauser & Goldberger (1971) MIMIC model _____	M-6
[M.2.5]:	Latent Growth Path Model from McArdle & Epstein (1987) _____	M-7
[M.5.1]:	Basic Regression Model (using RAM notation) _____	M-15
[M.7.1]:	Spleen data example generated from a SAS file _____	M-22

Part I: An Introduction to RAMpath

1. The RAMpath Program
2. Running RAMpath Quickly
3. Using this Manual
4. Our Historical Development
5. Some Acknowledgements

I.1: The RAMpath Program

To be, or not to be -- That is the question
Whether 'tis nobler in the mind to suffer
the slings and arrows of outrageous fortune ...

(W. Shakespeare, 1603, Hamlet, Prince of Denmark)

This manual describes a computer program called RAMpath. This computer program allows users to organize, display, analyze, and save "linear structural equation models". The algebraic basis of these models comes from well-known work in simultaneous equation systems. During the last decade these models have become very popular in the behavioral sciences under acronyms such as LISREL or COSAN. Our computer program is slightly different -- Instead of a heavy concentration on algebraic techniques, we focus on computer graphics. RAMpath is a computer program about "slings and arrows".

RAMpath has several advantages for users interested in dealing with graphic displays of path analysis models.

- * RAMpath deals with the graphic display and analysis of models. This program allows the user to automatically display a high quality path diagram on a computer screen or a printer.
- * RAMpath uses new automatic routines to calculate and display the tracing rules of path analysis. These calculations can be useful in analyzing model expectations and teaching about path analysis.
- * RAMpath has an easy way to define a path model using flexible script-based language. All RAMpath functions follow a mixture of C language conventions and simple commands.

RAMpath merges standard techniques in computer graphics (see the references by Boker) and structural equation modeling using RAM notation (see the references by McArdle). RAM theory is a simple way to define and use structural equation models as path diagrams. These principles are described in this manual in some detail.

Many researchers now find that these simple path analysis displays of simultaneous equations help communicate their overall analysis plan and

the numerical results. Graphic displays are becoming more and more important as models increase in complexity. We think path diagrams have not yet been used effectively in data organization and analysis. This is partly because there have been no widely available computer programs which easily and accurately draw path diagrams. There do not seem to be any computer programs which plot the tracing rules of path diagrams. RAMpath was designed to help users create high quality path analysis diagrams.

This version of RAMpath does not allow the user to estimate parameters from data or to evaluate model fit. We have released this version of RAMpath because we think there is an immediate need for an easy-to-use computer program for graphics of structural models. We do not ignore the estimation problem -- The list structure and language of this program is identical with our compatible estimation program (previously termed RAMit), and we also show how we can link up with other estimation programs such as LISREL and SAS/CALIS. RAMpath is designed to do one thing and do it well -- display and print path analysis diagrams.

I.2: Running RAMpath Quickly

The current RAMpath diskette enclosed with this manual includes:

(1) RAMPATH.EXE

The MSDOS executable code.

(2) *.RAM files

Several examples of input script files. These files are ASCII text files that can be edited with your favorite text editor. These files are read by RAMpath and used to create the path diagrams. You can try RAMpath by putting the RAMpath disk in the drive and typing RAMPATH EXAMPLE. This will read the file EXAMPLE.RAM and display an example path diagram.

(3) *.FON files

Several font files needed for different kinds of screen displays.

These files are not copy-protected, so all files can be transferred to a hard disk by using the MSDOS copy command.

RAMpath runs in batch mode from an input file stored on a disk. The programming language that is required is intended to be simple, straightforward, and potentially forgiving. These commands can be placed in the input file at any time and the order may or may not be important. A full list of commands is given in Table [1] on the command card enclosed here.

Assume we have an input file called "EXAMPLE.RAM". To run this program in MSDOS the user would enter:

```
>rampath example
```

and the RAMpath program reads this input script file line by line. This program then produces two kinds of output -- screen output and text output. The screen output can be saved in a postscript file called "EXAMPLE.PS", and the output automatically is saved in a file called "EXAMPLE.OUT". This output file gives a detailed history of the commands read, lists any errors found, and displays any output requested. The same basic procedure works with any path analysis problem.

I.3: Using this Manual

The rest of this manual is intended to provide more details and examples about these topics. All users should be able to read through each section, but these details may not be necessary for every applications.

- * The RAMpath Card -- The first three tables presented here tell most of the story.

Table [1] gives a list of all current RAMpath commands.

Table [2] gives a list of all RAM algebraic notation.

Table [3] gives a list of all RAM graphic notation.

- * "Part P: Programming with RAMpath" -- In this section we introduce some basic concepts in path models and show how we can use these within RAMpath. We discuss four main sections of the program -- INPUT, DRAW, OUTPUT, and SET. We discuss the drawing of RAMpath diagrams and the tracing rules of path analysis. We also illustrate a few advanced uses of RAMpath without too much detail. This section is most useful for beginners.

- * "Part M: Modeling with RAMpath" -- In this section we give a few illustrations and expand upon the basic algebraic and graphic concepts of RAM theory for path modeling. We also introduce additional topics, like matrix algebra representations and statistical estimation. We define all terms used in the program, but advanced users will be familiar with most concepts defined here.

- * "Part D: Dictionary of Terms" -- In this section we provide an alphabetical list of all RAMpath input commands. All program functions and options are listed here, and simple examples are presented to describe their usage. This reference section should be useful to all users.

I.4: Our Historical Development

The ideas expressed in this manual and program bring together the results of many other researchers in these areas. In the Fall of 1978 both authors of this program were working in Denver Colorado on similar projects -- the graphical display of mathematical and statistical models for data. The RAMpath program is the result of our specific collaboration.

At this time Jack McArdle was doing post-doctoral research on psychometric methodology. He was working on the use of the relatively new but popular ACOVSM, COFAMM, LISREL, and COSAN computer programs to analyze psychological data. After using up hundreds of hours of computer (and personal) time, McArdle finally figured out that all of these programs had a few common unifying graphic principles. This immediately meant that the algebra was not as complicated as it was usually presented and, correspondingly, the use of structural equation computer programs could be considerably simpler than the manuals suggested. McArdle presented these initial results at several professional meetings in 1978 and 1979. McArdle, together with the needed assistance of John Horn and Rod McDonald, developed various versions of what has now become known as RAM theory. The fundamental principles of RAM theory are now commonly used by researchers in the field and incorporated into several other structural modeling programs (e.g., CALIS, EzPath, etc.).

At the same time, Steve Boker was working hard (in his basement) on the creation of several new computer programs for the new world of computer graphics. Not surprisingly, Boker had also found some common

unifying principles of computer graphics, and he used these to develop a series of computer programs -- GRAPHTRIX, FONTRIX, and PRINTRIX. Both authors seemed to recognize that the other's work was eclectic, scattered, and potentially crazy, but also quite possibly correct and useful. So, what started as a chance occurrence, turned into a long lasting working relationship and friendship. The results of some of these unifying efforts are described in this manual. This is the first release of our decade long project.

This program does only one thing -- *RAMpath displays and prints path analysis or structural model based diagrams*. We think this program has a few advantages over all other current structural modeling programs. First, RAMpath deals with the graphic display and analysis of models. This program allows the user to automatically display a path diagram on a computer screen or a printer. Second, we also have developed automatic routines to calculate and display the tracing rules of path analysis. Third, and no less important, RAMpath allows freeform input of a model in a flexible script-based language (we call it the "MC"). This language adds a great deal to the ease of drawing a path diagram on a computer.

RAMpath is a personal statement. The concepts of path analysis presented here are original studies and developments of the first author (McArdle). The concepts of computer graphics and the C-code presented here were original developments of the second author (Boker). We are united in our view that RAMpath is "both necessary and sufficient", and we hope this program expresses this view. We have released this graphic drawing version because we found it to be valuable in our own work -- We now use this version of RAMpath to draw diagrams of our models, and we think this helps us clarify and refine our modeling ideas. We hope it can be as useful for you as well.

I.5: Some Acknowledgements

This work has been aided by many other people around the world during the last ten years. Most notable here is John L. Horn, then at the University of Denver and now at the University of Southern California. John Horn brought the two authors together, generously supported this project in all phases of its development, and he is the silent third author of RAMpath.

Other scientists who have kindly supported our work include John Nesselroade, John Loehlin, Rod McDonald, Jan-Bernd Lohmoeller, Hill Goldsmith, Michael Browne, Wayne Velicer, Bob Cudeck, Harry Gollob, Bill Meredith, Stan Mulaik, Jim Steiger, Keith Widaman, Jim Connell, Dave Rindskopf, and Mike Neale. Other colleagues who have helped with this work include Greg Smith, Duane Buhrmester, Harry Gollob, Chip Reichardt, Bill Gardner, Mark Aber, David Epstein, Aki Hamagami, and Jim Pisano. We would like to extend a special thanks to Carol Prescott whose careful reading of the manuscript was invaluable. Our work was also supported by members of the Data Transforms team, including Bob VanArsdale, Glenn Searfoss, and Margaret Devere. We wish to thank Valerie Larsen without whose encouragement and tolerance the software code could not have been completed. The McArdle children have provided both stimulus and response for the continuing development of simpler versions of RAM theory. We are also indebted to many other researchers and friends too numerous to list here.

This is an experimental program (and manual) and we expect it will be undergoing many changes in the near future. We'd very much appreciate your comments and questions. Please help us by sending your critical comments, other suggestions, or favorite RAMpath examples to:

Data Transforms, Inc.
616 Washington Street
Denver, CO 80203
Phone: 303-832-1501
Electronic-mail address : rampath@datran2.uunet.uu.net

Thanks for your assistance and support.

Part P: Programming with RAMpath

1. A simple example of RAMpath
2. RAMpath language and commands
3. Drawing a path diagram
4. Tracing the rules of path analysis
5. Matrix input and output
6. Additional examples on disk

P.1: A Simple Example of RAMpath

Simple path analysis displays of simultaneous equations help communicate the overall analysis plan and the numerical results. Graphic displays become more and more important as the model increases in complexity. These path diagrams can be easily drawn and evaluated with RAMpath.

We first show how RAMpath works in a decidedly simple case. Let us assume we have created a disk file named "sqrt.ram" with the following list of entries:

File [P.1.1]: A simple input file for RAMpath

```
/* ----- */
/* This is an input script file called Sqrt.RAM */
/* ----- */

set_title = "A very simple first example of RAMpath";
input_model variable(1) = manifest;
input_model variable(2) = latent;
input_model arrow(1,2) = 1.414;
input_model sling(2,2) = 1;
draw_model;
output_model;
/* ----- */
```

RAMpath runs in batch mode from an input file stored on a disk. To run this program in MS-DOS the user would type:

```
>rampath sqrt
```

and the RAMpath program reads this script file. This program then produces two kinds of output -- screen output and text output. The first output of importance is the diagram, and this is shown below (see Figure [P.1.1]).

After running this simple program we also obtain printed output stored in a textfile. This file gives a detailed history of the commands read, any errors observed, and a listing of the internal model list. For this example we obtain:

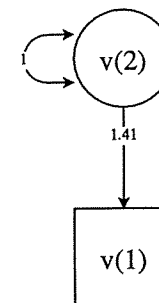


Figure [P.1.1] A very simple first example of RAMpath

File [P.1.2]: An example of an output textfile generated by RAMpath

```
RAMpath (c) 1980 Jack McArdle & Steve Boker
RAMpath input file = sqrt.ram
```

```
DRAW_MODEL based on current options
Calling the RAPGRAF function
Following output based on current options
```

```
Current title = 'A very simple first example of RAMpath'
Current MODEL includes 4 parameters
```

Name	Matrix	Row	Column	Value	Error
Filter	1	1		1.00000	0.00000
Filter	1	2		0.00000	0.00000
Arrow	1	2		1.41400	0.00000
Sling	2	2		1.00000	0.00000

```
-----
End of input file
Thank you for using RAMpath
```

This simple example illustrates many features of our program language. In the input of File [P.1.1] above, all lines that start with "/"

and end with “*/” are comments. Commands always begin with a command keyword and all commands end with a “;”. Spaces, tabs and carriage returns do not affect the command so the user has considerable latitude in formatting the appearance of the command lines. The “set_title” command allows us to input a title (as a quoted string). The “input_model” command allows the user to describe the structural equation model as a list of variables and parameters. Here we enter four input model statements. The “draw_model” command produces a path diagram on the screen. The “output_model” command produces a list of the current model on the output file. This language will be presented in further detail in the this section and in the Dictionary at the end of this manual.

The resulting diagram of Figure [P.1.1] can also be printed on postscript printers or printers that support a graphics Print Screen. The general translation from input words to output pictures comes easily because RAMpath always uses a “list” structure like the one output above in File [P.1.2].

This example is obviously intended to produce a very simple picture. In fact, this is one of the simplest possible models we can use with RAMpath. We can use this model to describe (and calculate) the square-root of a number (i.e., the sqrt(2) in this case). But we start with this simple example to demonstrate one key point -- No matter what complexity we create in a model, the same basic language and logic is used for the input and the output of any path model.

P.2: RAMpath language and commands

The programming language used in RAMpath is intended to be simple, straightforward, and potentially forgiving. All commands are input using a “script” language of the same general form:

```
command_name = numbers (and/or) "string" ;
```

All commands (1) begin with a “command_name”, (2) have optional numbers or quoted strings, and (3) end in a semicolon. Extra spaces, equal signs, and parentheses are not necessary but they can be used to clarify the meaning of the command line.

Some specific examples of command lines are:

```
set_title = "This is the first title line";
set_print = more;
set_width = 7;
input_model variable(1) = manifest;
output_model;
```

Comments may be placed inside the input text at any time using standard C-convention comment labels “/*” and “*/”. These are mainly used to document the input stream, but creative users may find comments useful in other ways. For example:

```
/* This is a comment */
```

```
/* So is
   this */
```

Comments do not effect the input program in any way, but comments cannot be nested inside one another.

Most RAMpath program input files have five main sections: INPUT, OUTPUT, DRAW, SET, and GENERAL commands.

- * INPUT commands -- These are a set of commands used to get a model into the program. Input of a model can come in the form of elements, lists, or matrices.
- * DRAW commands -- These are a set of commands used to draw a path diagram of a model. This is done automatically by the program by evaluating the patterning in the current model list. The manifest variables are drawn as “squares” and the latent variables are drawn as “circles”. The arrow parameters are drawn as “one-headed” arrows and the “slings” are drawn as “two-headed” arrows. The program can also automatically generate a graphic picture of the “tracing rules of path analysis” for any pair of variables in the diagram. The program includes many options for alternative drawings, including the use of triangles, dashed-boxes, line-widths, color, and so on. These pictures can also be save in a postscript file for high quality printing.
- * OUTPUT commands -- These are a set of commands used to output a model from the program. RAMpath can output the current model as a list or as a set of matrices. RAMpath can also automatically generate and output the “tracing rules of path analysis” for any pair of variables

in the diagram. The program can save the model to an ASCII textfile that can be reinput, edited, or used by another program.

- * SET commands -- These are a set of commands used to change the model defaults. Users can redefine various input, output, or drawing options.
- * GENERAL -- The RAMpath program has an easy to use command language which allows the user a variety of options for model definitions on input, output, and comments. These commands include comments, aliases, and #DEFINE.

P.3: Drawing Path Diagrams

In order to move to more complex models we need to be more specific about some formal distinctions. To draw a path diagram using RAM theory we need to make three required graphic distinctions:

- (G1) Variables v are different from Parameters P -- Any variable $v(i)$ will be drawn as a labeled point or VERTEX, and any parameter $p(ij)$ will be drawn as a labeled line or EDGE between variables $v(i)$ and $v(j)$.
- (G2) Manifest g variables are different from latent h variables -- Any observed, manifest, or given variable $g(i)$ will be drawn as a labeled SQUARE (or rectangle), but any unobserved, latent, or hidden variable $h(j)$ will be drawn as a labeled CIRCLE (or ellipse).
- (G3) Asymmetric A parameters are different from Symmetric S parameters -- Any asymmetric or directed regression parameter $a(ij)$ will be drawn as a line with one arrowhead, an ARROW, pointing from one variable $v(j)$ to one variable $v(i)$, but any symmetric or undirected expectation parameter $s(xy)$ will be drawn as a line with two arrowheads, a SLING, pointing towards and connecting the variables $v(x)$ and $v(y)$.

The diagram of Figure [P.3.1] is a prototypical example of a contemporary path diagram. This model includes six observed variables labeled $v(1)$, $v(2)$, $v(3)$, $v(4)$, $v(5)$, and $v(6)$, as well as one unobserved variable labeled $v(7)$. The first three variables all have slings representing variance $W(i)$ and correlation $R(i,j)$. These three variables

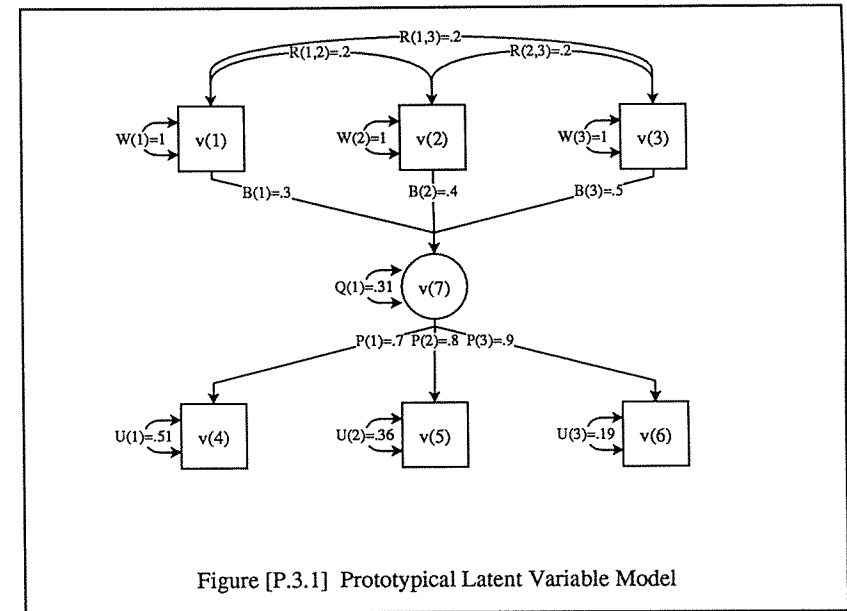


Figure [P.3.1] Prototypical Latent Variable Model

also have arrows representing regression effects $B(i)$ on unobserved $v(7)$ with a disturbance variance labeled Q . Also $v(7)$ has direct effects on the other three observed variables with factor loadings $P(j)$ and unique variances $U(j)$. This patterning of the one-headed arrows and two-headed slings allows the interpretation of variable $v(7)$ as a "mediational" variable between input set $v(1-3)$ and output set $v(4-6)$. The numerical values listed here are somewhat arbitrary, but they can be used for later calculations (see Matrix section below).

Models such as this one are common in contemporary path analysis treatments of external and internal validity (e.g., Kenny, 1980, p.180; Bagozzi et al, 1982; McArdle & Prescott, 1990). Of course, this model might just as well be considered as a specific configuration of squares and circles variables with one-headed arrows and two-headed slings. Either way the model input and output is identical.

Without any further definitions we can list a few simple steps required in drawing a RAMpath diagram.

1. Draw the picture by hand. This initial scribbling does not need to be neat, but it will help get started.

File [P.3.2] : A RAMpath output text file for the latent variable model

Current title = 'RAMpath (c) 1986-1990 by J.J.McArdle & S.M.Boker'

Current MODEL includes 23 parameters

Name	Matrix	Row	Column	Value	Error
Filter	1	1	1	1.00000	1.00000
Filter	2	2	1	1.00000	1.00000
Filter	3	3	1	1.00000	1.00000
Filter	4	4	1	1.00000	1.00000
Filter	5	5	1	1.00000	1.00000
Filter	6	6	1	1.00000	1.00000
Filter	1	7	1	0.00000	1.00000
Arrow	7	1	2	0.30000	1.00000
Arrow	7	2	3	0.40000	1.00000
Arrow	7	3	4	0.50000	1.00000
Arrow	4	7	1	0.70000	1.00000
Arrow	5	7	2	0.80000	1.00000
Arrow	6	7	3	0.90000	1.00000
Sling	1	1	2	1.00000	1.00000
Sling	2	2	3	1.00000	1.00000
Sling	3	3	4	1.00000	1.00000
Sling	1	2	3	0.20000	1.00000
Sling	1	3	4	0.20000	1.00000
Sling	2	3	4	0.20000	1.00000
Sling	4	4	5	0.51000	1.00000
Sling	5	5	6	0.38000	1.00000
Sling	6	6	7	0.19000	1.00000
Sling	7	7	8	0.31200	1.00000

This output above is a list of the internal matrix representation of the model. In this list every row represents a non-zero parameter and each column is a specific characteristic of that parameter. For example, the third row represents a parameter in matrix 0 (the filter) which is at row=3 and column=3 with a value=1. The eighth row represents a parameter in matrix 1 (the arrows) at row=7 and column=1 with a value=.3. The last row represents a parameter in matrix 2 (the slings) at row=7 and column=7 with a value of .312. All model parameters are described in this same way, and more details are listed under INPUT_MODEL in the dictionary.

There are many more examples of input and output given on the RAMpath disk. For more details, see the section on Examples below.

P.4: Tracing Through a Path Diagram

The traditional rules of path analysis tracing are clearly presented by Wright (1985; also see Li, 1975; Alwin & Hauser, 1975; Fox, 1980). These rules are detailed in the Modeling section of this manual. A complete set of expected relations between variables can be reconstructed using some form of graphic tracing rules, and these interpretations are well known to experienced path analysts. But these interpretations are confusing for those less experienced, and they are difficult to teach. "Is this a valid tracing?" or, "Why trace it that way?" or, "Are there really any tracing rules?" or, "Why does tracing work at all?" are common, and reasonable questions. The answers to some of these questions can come from using the OUTPUT_BRIDGES and DRAW_BRIDGES commands.

RAM theory can be used to summarize all tracing principles into three required "tracing rules":

- (G4) Effects = E_p = a specific sequence of E matrix elements which can be seen as a "tracing along a sequence of arrows." The sequence labeled $E_{p[i,j]}$ indicates an action from variable $v(i)$ to variable $v(j)$ along p arrows.
- (G5) Bridges = B = a specific sequence of A and S matrix elements which can be seen as "an action backwards, one and only one turn on a sling, and an action forward". The bridge labeled as $B[l<-2<->2->3] = E_{p[1,2]} S_{(2,3)} E_{q[2,3]}$ is an action of length p from variable $v(1)$ backward to $v(2)$, followed by a turn around on the sling between variable $v(2)$ and $v(3)$, and finally followed by an action of length q from variable $v(2)$ to $v(3)$.
- (G6) Connections = C = the total sum of all bridges between two variables. This can be seen as an accumulation of all the ways to trace or travel in the diagram from a specific set of variables.

The RAMpath program has several facilities for dealing with this path calculation. Initially the program maps out the entire set of connections into a list structure. Next, we display the connection among any two variables in the model as a subgraph. Each of these subgraphs displays only one element in the total correlation between a pair of variables; Because of this function, we term this subcorrelation as a *bridge*.

As an example, let us say we are interested in the reason why variables $v(1)$ and $v(4)$ in the previous latent variable model have a non-zero covariance. To deal with this problem the following commands are added to the previous example.

```
OUTPUT_BRIDGES = (1, 4);
DRAW_BRIDGES = (1, 4);
```

The OUTPUT_BRIDGES command lists all the specific information.

File [P.4.1] : The text output produced by OUTPUT_BRIDGES

```
OUTPUT_BRIDGES read with 2 numbers
OUTPUT_BRIDGES based on current options
Calling the RAP_RXPGRAF function
```

```
*****
ARROWS between VARIABLE 1 and VARIABLE 4
```

Index	Value	ARROW Description
1)	0.210000	1->7->4

```
Listed ARROWS = 1
Total ARROWS = 18
```

```
*****
Listing of Components of Cross Products (BRIDGES)
```

```
BRIDGES between VARIABLE 1 and VARIABLE 4
```

Index	Value	BRIDGE Description
1)	0.210000	4<-7<-1<->1
2)	0.058000	4<-7<-2<->1
3)	0.070000	4<-7<-3<->1

```
=====
0.338000
```

```
Listed Bridges = 3
Total Bridges = 133
```

The DRAW_BRIDGES command reinterprets this notation as a set of three different subgraphs between variables $v(1)$ and $v(4)$, and these are produced as the following picture. (See Figure [P.4.1])

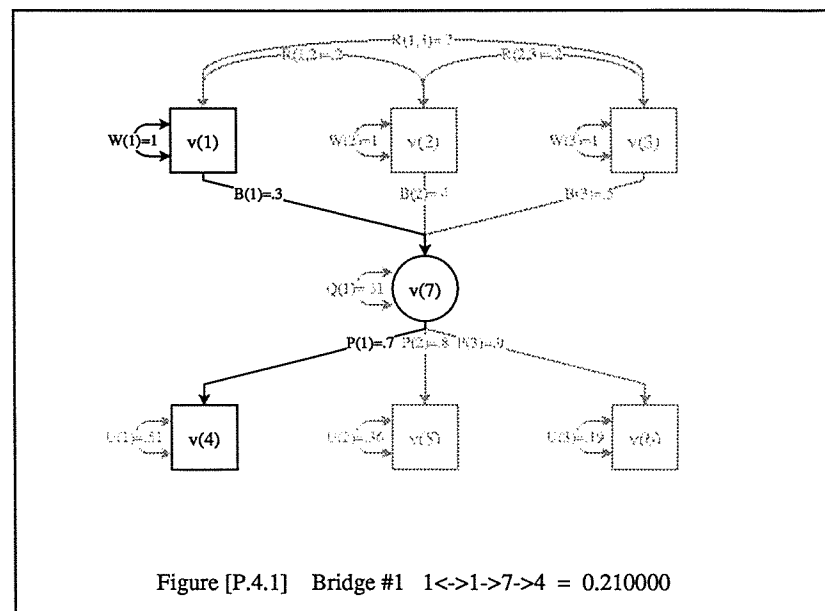


Figure [P.4.1] Bridge #1 1<->1->7->4 = 0.210000

The first tracing starts backwards from variable $v(4)$ towards variable $v(7)$, continues backward from $v(7)$ to $v(1)$, and turns around at variable $v(1)$. Because the end-variable desired, $v(1)$, has been reached we need not go further forward. The numerical value of this bridge is the product of the parameters used in this specific action, so $(.7 \times .3 \times 1.0) = .210$. This bridge is symbolized as "4<-7<-1<->1". Of course, the correlation of $v(4)$ and $v(1)$ is a symmetric coefficient, so "4<-7<-1<->1" also equals "1<->1->7->4". This first tracing is a typical graphic example of nonlinear structural model expectations.

The other two bridges in this model represent other components in the correlation of $v(4)$ and $v(1)$. These would be automatically drawn by the program, but this output is not included here. The second bridge, 4<-7<-2<->1 = .056 is similar but it uses a "less direct" tracing through variable $v(2)$. The one and only one sling used in this bridge is the correlation of $v(2)$ and $v(1)$. The third bridge, 4<-7<-3<->1 = .070, is similar in form.

In all cases, the sum of all separate bridges equals the total connection among the variables. In this example $1 \text{ 4<-7<-2<->1} + 4\text{-7<-2<->1} +$

$4<-7<-2<->1] = C(4,1)$, so $[.210 + .056 + .070] = .336$, which is the numerical value for the expected correlation between $v(1)$ and $v(4)$. The interpretation of this expected correlation can become clearer. The separate bridges each contribute a percentage to the expected correlation, and the first bridge is clearly the largest here. Perhaps the most relevant aspect of this model is that the latent variable $v(7)$ is used in all connections between $v(1)$ and $v(4)$.

A more complex calculation is required for the contribution of the model parameters to the variance of a single variable. For example, we can add the following commands to the previous example.

```
OUTPUT_BRIDGES 4 4;
DRAW_BRIDGES 4 4;
```

The OUTPUT_BRIDGES command lists all the specific information.

File [P.4.2]: Text output for the OUTPUT_BRIDGES of the Variance term.

```
*****
ARROWS between VARIABLE 4 and VARIABLE 4
```

Index	Value	ARROW Description
-------	-------	-------------------

```
Listed ARROWS = 0
Total ARROWS = 15
```

```
*****
```

Listing of Components of Cross Products (BRIDGES)

```
BRIDGES between VARIABLE 4 and VARIABLE 4
```

Index	Value	BRIDGE Description
1)	0.044100	4<-7<-1<->1->7->4
2)	0.078400	4<-7<-2<->2->7->4
3)	0.011780	4<-7<-2<->1->7->4
4)	0.011780	4<-7<-1<->2->7->4
5)	0.122500	4<-7<-3<->3->7->4
6)	0.014700	4<-7<-3<->1->7->4
7)	0.014700	4<-7<-1<->3->7->4
8)	0.019800	4<-7<-3<->2->7->4
9)	0.019800	4<-7<-2<->3->7->4
10)	0.510000	4<->4
11)	0.152880	4<-7<->7->4

```
=====
1.000000
```

```
Listed Bridges = 11
Total Bridges = 133
```

In this example, there are eleven separate components to the variance of variable $v(4)$. These components follow the usual algebraic rules for any regression model, but here the model is complex and some of the variables are unobserved.

The increasing complexity of a model makes some bridges hard to visualize and hard to calculate. Still, all bridges follow the same general pattern: Any bridge is formed by (1) tracing backward on some (>0) arrows, (2) turning direction on one and only one sling, and (3) tracing forward on some (>0) arrows. While all tracings require *one and only one sling* the number of arrows can be *greater than or equal to zero*. The tracing is identical in the opposite direction but the rules above do not change at all. The tracing of *any* bridge follows these simple graphic interpretations. These interpretations can help the path analyst better understand the meaning of the placement of a specific parameter in the context of the other model parameters.

P.5: Matrix input and output

Another way to input and output a model list is based on matrix algebra. Matrix notation is useful because it provides a direct link to the available estimation programs and because it is a compact way of defining a model. This algebra is described in some detail in the next section, but we can overview these ideas here.

First we define all variables as:

- (A1) manifests = g = the $(l \times p)$ matrix of manifest (observed) variables,
- (A2) latents = h = the $(l \times q)$ matrix of latent (unobserved) variables, and
- (A3) variables = $v = [g : h]$ = the $(l \times t)$ matrix of total variables,

Next we define three parameter matrices:

- (A4) Filter = F = the $(p \times t)$ binary matrix which "filters" the manifest from the latent variables,

(A5) Arrows = A = the $(t \times t)$ asymmetric parameter matrix of all "one-headed arrows", so $v = v A + u$, and

(A6) Slings = S = the $(t \times t)$ symmetric parameter matrix of all "two-headed arrows" or "slings", so $S = \Sigma \{ u u' \}$.

These definitions allow us to calculate three model matrices:

(A7) Effects = $E = (I - A)^{-1}$ the $(t \times t)$ asymmetric matrix of total regression or reduced form coefficients,

(A8) Connections = $C = E S E'$ = the $(t \times t)$ symmetric matrix of expectations among all variables v , and

(A9) Expectations = $X = F C F'$ = the $(p \times p)$ symmetric matrix of expectations among manifest variables m .

In simplest terms, the RAM matrices A , S , and F can be input directly to the program, and the matrices E , C , and X can be calculated and output directly from the program.

Matrix input to RAMpath is only one of the ways to build up the elements of the list. However, this can be convenient, so we use it often. A simple example of the language syntax follows:

```
/* this is a simple example of matrix model input */
```

```
input_model_matrix
  arrows(5,5) =
    [ 0 0 0 0 1
      0 0 0 0 1
      0 0 0 0 1
      0 0 0 0 0
      0 0 0 0 0 ];
```

```
/* and this is equivalent to a list input where */
```

```
input_model
  arrow(1,5)=1;
  arrow(2,5)=1;
  arrow(3,5)=1;
```

The matrix input approach requires us to define the matrix, the

number of rows and columns, and the values. Only the non-zero values are interpreted as model parameters and the zero elements are ignored.

The RAMpath input listed below gives a brief example of matrix algebra input and output for the latent variable path model used earlier (see Figure [P.3.1]).

File [P.5.1] RAMpath matrix input of the latent variable path model

```
/*
A step-by-step matrix input of a latent variable path
model
*/

/* Step (1): Enter the manifest variables */

input_model_matrix
  filter (8,7) =
    [ 1 0 0 0 0 0 0
      0 1 0 0 0 0 0
      0 0 1 0 0 0 0
      0 0 0 1 0 0 0
      0 0 0 0 1 0 0
      0 0 0 0 0 1 0
      0 0 0 0 0 0 1 ];

/* Step (2): Enter all the one headed arrows */

input_model_matrix
  arrow (7,7) =
    [ 0 0 0 0 0 0 0
      0 0 0 0 0 0 0
      0 0 0 0 0 0 0
      0 0 0 0 0 0 .7
      0 0 0 0 0 0 .8
      0 0 0 0 0 0 .9
      .3 .4 .5 0 0 0 0 ];

/* Step (3) Enter the two-headed slings */

input_model_matrix
  sling (7,!) =
    /* the ! means the matrix is lower triangular */
    [ 1
      .2 1
      .2 .2 1
      0 0 0 .51
      0 0 0 0 .36
```

```

O O O O O .19
O O O O O .312 1;

```

/* Step (4): Output the results */

```

draw_model;
output_model;
output_model_matrix;

```

This program again produces two kinds of output -- screen output and text output. The diagram produced by this input is the same as the diagram produced by the list input. The main difference here comes in the contents of the *.OUT file. This file gives a detailed history of the commands read as well as a listing of the internal model matrices.

File [P.5.2] : RAMpath matrix output for the latent variable path model

OUTPUT_MODEL_MATRIX read with 0 numbers

All model matrices output based on current parameter values

F = Fixed FILTER Matrix = (8 x 7)

```

1.0000  0.      0.      0.      0.      0.      0.
0.      1.0000  0.      0.      0.      0.      0.
0.      0.      1.0000  0.      0.      0.      0.
0.      0.      0.      1.0000  0.      0.      0.
0.      0.      0.      0.      1.0000  0.      0.
0.      0.      0.      0.      0.      1.0000  0.

```

A = Asymmetric ARROW Matrix = (7 x 7)

```

0.      0.      0.      0.      0.      0.      0.
0.      0.      0.      0.      0.      0.      0.
0.      0.      0.      0.      0.      0.      0.
0.      0.      0.      0.      0.      0.      0.7000
0.      0.      0.      0.      0.      0.      0.8000
0.      0.      0.      0.      0.      0.      0.9000
0.3000  0.4000  0.5000  0.      0.      0.      0.

```

S = Symmetric SLING Matrix = (7 x 7) Symmetric

```

1.0000
0.2000  1.0000
0.2000  0.2000  1.0000
0.      0.      0.      0.5100
0.      0.      0.      0.      0.3600

```

```

0.      0.      0.      0.      0.      0.1900
0.      0.      0.      0.      0.      0.      0.3120

```

C = CONNECTION Matrix of Manifest & Latent Connections = (7 x 7) Symmetric

```

1.0000
0.2000  1.0000
0.2000  0.2000  1.0000
0.3380  0.3920  0.4480  1.0000
0.3840  0.4480  0.5120  0.5800  1.0000
0.4320  0.5040  0.5760  0.6300  0.7200  1.0000
0.4800  0.5800  0.6400  0.7000  0.8000  0.9000  1.0000

```

X = EXPECTED Matrix of Manifest Connections = (8 x 8) Symmetric

```

1.0000
0.2000  1.0000
0.2000  0.2000  1.0000
0.3380  0.3920  0.4480  1.0000
0.3840  0.4480  0.5120  0.5800  1.0000
0.4320  0.5040  0.5760  0.6300  0.7200  1.0000

```

These last two matrices are model expectations calculated from the algebraic relationships.

P.6: Additional Path Diagram Examples

A variety of other examples of RAMpath input scripts are included as files on the disk. In order to save room, we do not print or detail these examples. But we do encourage all users to examine each one of these examples in sequence. Each one of these files illustrates a slightly different aspect of the RAMpath program.

In a first set of examples we present a few theoretical models. These examples illustrate RAMpath input and output of:

SQRT.RAM	A one parameter model for the square root problem (Figure [P.1.I])
REGRESS.RAM	A multiple regression model
PATH.RAM	A typical path analysis model
FACTOR.RAM	A typical factor analysis model
LATENT.RAM	A typical latent variable path model (Figure [P.3.I])
CANPATH.RAM	A sequence of nested latent variable path models

SIMPLEX.RAM	A longitudinal simplex path model
FEEDBACK.RAM	A simple feedback model
TWINA.RAM	A model for twin pairs
FROMSAS.RAM	The output of SAS/CALIS useful for RAMpath (Figure [M.7.1])

A second set of examples comes from the published literature in this area. These examples illustrate RAMpath input and output of:

DUNCAN.RAM	The latent variable non-recursive model of Duncan & Haller
HAUSER.RAM	The Hauser & Goldberger MIMIC model (Figure [M.2.4])
HORN.RAM	The Horn model of the Woodcock-Johnson Cognitive scales
JORESKOG.RAM	The Longitudinal Factor Model of the SAT by Joreskog
LAWLEY.RAM	The Lawley & Maxwell factor analysis model
MCARDLE.RAM	The latent growth curve model of McArdle (Figure [M.2.5])
MCDONALD.RAM	The McDonald invariant factors longitudinal model
SPEARMAN.RAM	The single factor model of Charles Spearman (Figure [M.2.1])
MCAHORN.RAM	The McArdle & Horn model of the WAIS
WIDAMAN.RAM	The Multi-Method Multi-Trait model of Widaman
WRIGHT.RAM	The original path model of Sewell Wright (Figure [M.2.2])

Part M: An Overview of Structural Modeling

1. Introduction to structural modeling
2. Structural equation models and path diagrams
3. The algebraic principles of RAM theory
4. The graphic principles of RAM theory
5. RAM versus traditional path diagrams
6. Statistical Principles
7. Drawing diagrams from SAS and LISREL

M.1: Introduction to Structural Modeling using RAMpath

In this section we will expand upon the basic concepts of path diagrams and structural equation modeling. This presentation is not intended to be a primer for these topics. Readers interested in these topics should examine the books by Loehlin (1989) and McDonald (1985). Instead we provide a few details about structural modeling that are consistent with our use of path analysis graphics in RAMpath. We recognize there are other alternative approaches to these problems. Users interested in LISREL notations should see the recent book by Hayduk (1989), and should also see McArdle & Hamagami (1989).

The graphic display of systems of equations was initially developed by Sewall Wright in what he called "path analysis" (1918, 1921, 1934, 1982). Without doubt, these path analysis displays have made these structural techniques more effective in communication and immediately available to a broad audience. These graphics can also be used to effectively communicate and program a variety of advanced algebraic models. In some cases these graphics have altered the way social science researchers think and communicate constructs and data.

Since the early 1970s there has been rapidly increasing interest in the use of linear structural equation modeling techniques in the social sciences. These techniques include many different kinds of parametric statistical models including the analysis of variance, regression analysis, factor analysis, and path analysis. One reason for increased interest is the availability of computer programs, such as the popular LISREL series (Joreskog & Sorbom, 1979, 1985) and other programs such as the less well-known COSAN programs (McDonald, 1978, 1979, 1985; Fraser, 1979). These structural equation methods have become popular because they are related to some simple graphical displays. In some ways these new computer programs have altered the way researchers analyze social science data.

The acronym RAM stands for "Reticular Action Model". This term was taken from Cattell (1966) and others who tried to describe behavioral phenomena as interlocking systems of variables and parameters. The algebraic development of RAM notation is fully expressed by McArdle & McDonald (1984), and this is based largely on the previous structural equation models of McDonald (1978) and Joreskog & Sorbom (1979). The graphic development of RAM is expressed by McArdle (1978, 1980), and these graphics are based largely on the previous path analysis models of

Wright (1918, 1921, 1934, 1982). First we will give some common examples of RAM graphics. Later we will describe the interplay between RAM algebra and RAM graphics.

M.2: Structural equation models and path diagrams

Many contemporary users of confirmatory factor analysis express their models as path diagrams. The first structural equation model of this kind was developed and fit by Charles Spearman (1904).

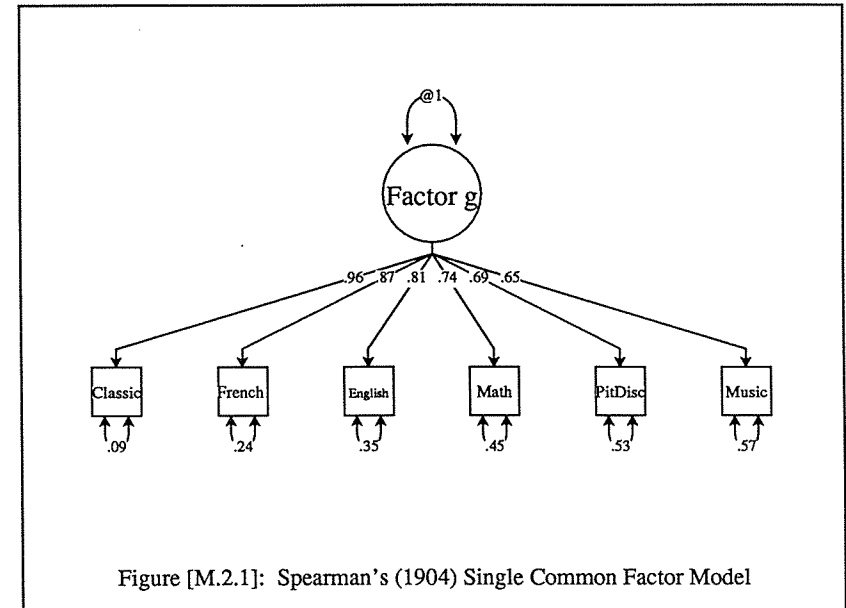
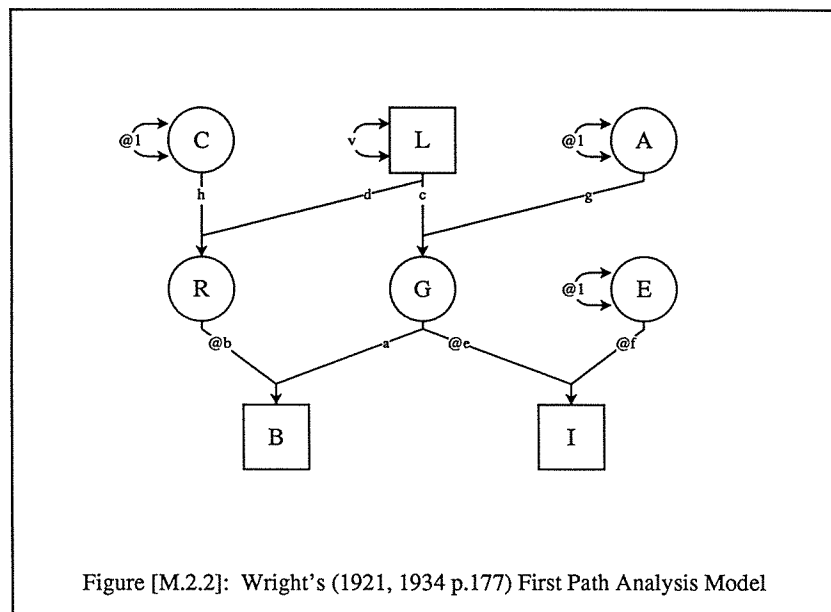


Figure [M.2.1]: Spearman's (1904) Single Common Factor Model

Spearman was interested in testing the idea that a single underlying factor was responsible for the observed scholastic abilities of young children. In this analysis Spearman used six observed (or manifest) variables, and one unobserved (or latent) variable labeled *g*. In these diagrams we draw the manifest variables as labeled squares and all latent variables as labeled circles. Spearman's *g* was assumed to be an ability factor which was general to many different ability tests. The one-headed arrows are used here to represent the factor loadings. The two-headed

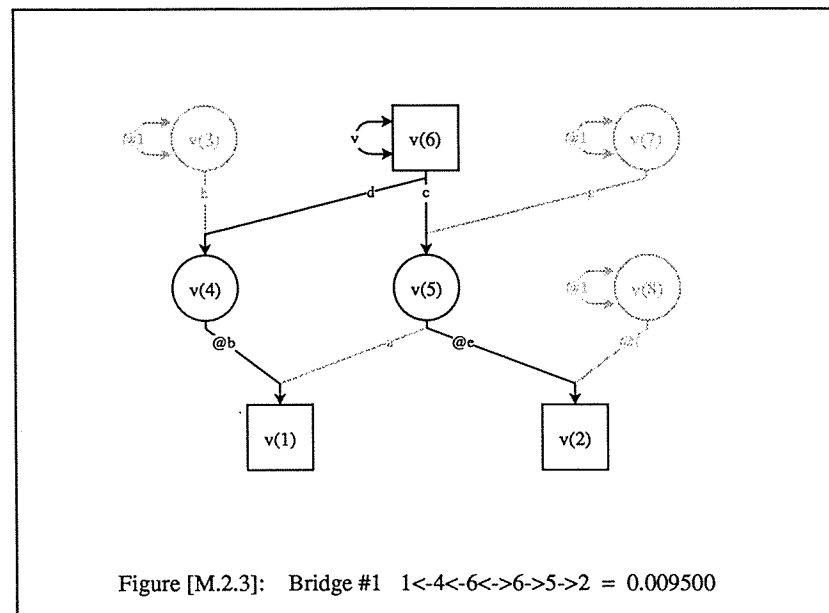
arrows are used to represent the variance of the factor g , and the unique variance of the variables. As we will detail later, any two-headed arrow represents variance that is not accounted for by the one-headed arrows pointing into a variable. At first this alteration from standard path diagramming is confusing. However, this simple device allows us to draw a compact diagram which retains an accurate matrix algebra. Spearman did not actually draw a path model, but we take the liberty of expressing his factor analysis equations in this way (see Figure [M.2.1]).

A second example comes from the original path analysis model of Sewell Wright. In this early work Wright (1918, 1921, 1934) examined alternative pathways for the birth weight of litters of guinea pigs. This first path model explicitly included both manifest and latent variables, and partial regression coefficients were drawn with "one-headed arrows" while correlation coefficients were drawn as "two-headed arrows" (see Figure [M.2.2]).



Wright recognized that these diagrams also have a second more complex function -- Path diagrams can be used to describe the "model expectations" of the statistical characteristics of data. For example, the

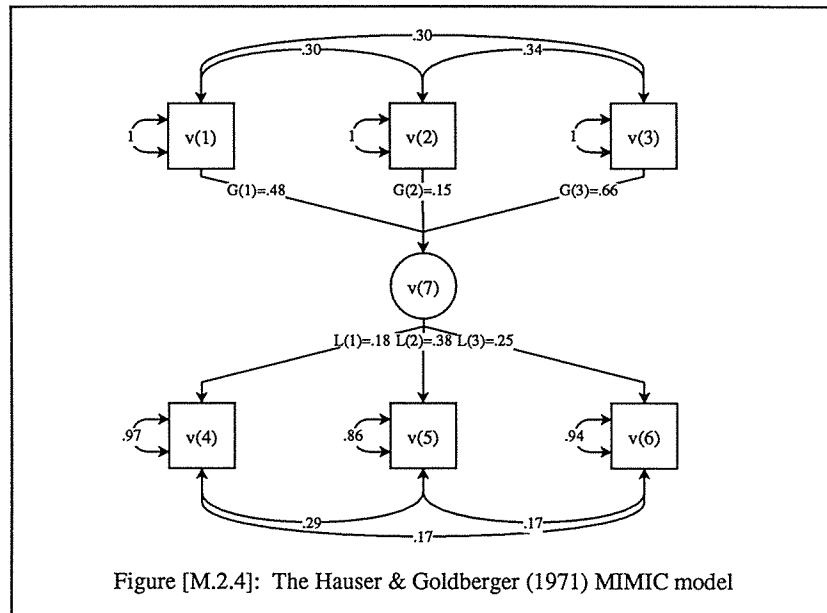
particular correlation between variable $v(1)$ and variable $v(2)$ in the model above may be implied as the sum of three products of the other model parameters. One of these three products are presented in the next picture below.



As we have stated, the rules that govern the particular combinations of sums and products can be derived from the matrix algebra, and these algebraic relations may also be expressed graphically as movements in a path diagram. The movements drawn here follow the well known "tracing rules of path analysis" defined quite succinctly by Sewell Wright (1921, 1982). These tracings are most useful in understanding the complexity and implications of a specific model for data. These tracing rules are most useful in decomposing complex models such as this one (see Figure [M.2.3]).

Path analysis diagrams were used in genetic research by Wright (1934) but they did not attract much attention in early psychological research, especially after the critique of Englehardt (1934). In fact, path analysis lay dormant in all behavioral science for about five decades until Duncan (1966) revived the methods with some applications in sociology.

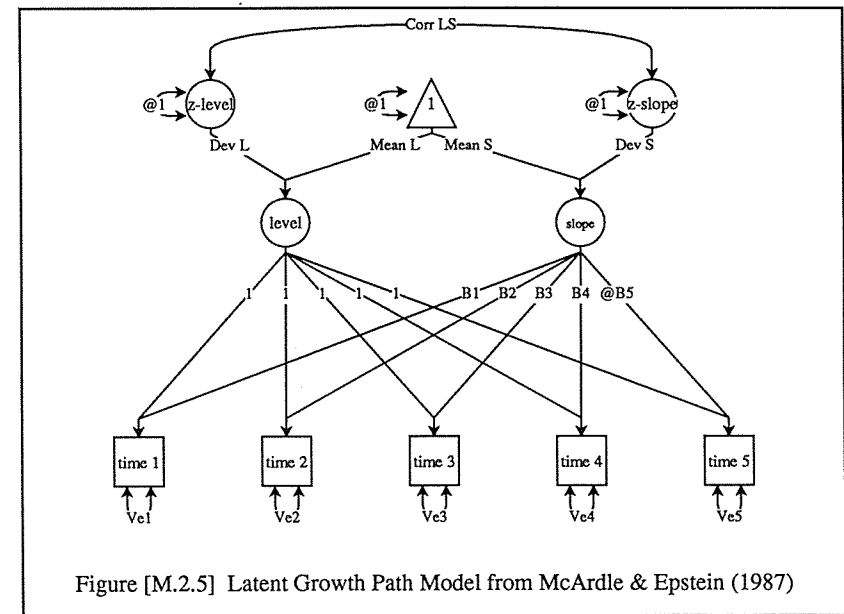
Relevant psychological applications of path analysis were subsequently proposed by Werts & Linn (1970) to represent true-score repeated measures and analysis-of-covariance models. The multivariate displays of Van de Geer (1971) and the sociological research of Hauser & Goldberger (1971) added technical devices for treating path analysis models. These presentations were among the first to detail a formal treatment of “unobservable” variables representing common factors within a path model (of course, see Wright, 1934).



The next diagram from Hauser & Goldberger (1971) is a useful example of another kind of latent variable path model. This model includes six observed variables labeled $v(1)$, $v(2)$, $v(3)$, $v(4)$, $v(5)$, and $v(6)$, as well as an unobserved variable labeled $v(7)$. This model was termed a “Causal model of socioeconomic status and participation.” In other work this model was referred to as a “Multiple Indicators and Multiple Causes” or MIMIC model. In this original display the observed variables were drawn as circles and the unobserved latent variable $v(7)$ was drawn as a square. The one-headed arrows here show that the $v(i)$ variables have regression-like effects $G(i)$ on unobserved $v(7)$ with zero disturbance, but also that $v(7)$ affects the $v(4)$, $v(5)$ and $v(6)$ variables with regression-like

effects $L(1)$, $L(2)$ and $L(3)$. This patterning of the one-headed and two-headed arrows allows the interpretation of variable $v(7)$ as a mediational variable between input set $v(1)$, $v(2)$ and $v(3)$ (e.g., social status variables) and output set $v(4)$, $v(5)$ and $v(6)$ (e.g., social participation variables). Drawings such as this one are common in contemporary path analysis treatments.

One final path analysis model comes from our own work. Figure [M.2.5] shows a model for longitudinal data called a “Latent growth curve” (by McArdle & Epstein, 1987; McArdle, 1988, 1989; McArdle & Anderson, 1989). In this theoretical model we assume the measurement of a single variable over time and we try to use latent variable models to structure the covariation over time and the mean change over time. Here we draw five observed time points (labeled “time 1” to “time 5”). We assume there are several underlying latent variables with effects on these observations. We assume the presence of (1) a “level” factor with unit weights to all time points, (2) a “slope” factor with $B(t)$ weights to all time points, and (3) an independent “error” factor (not drawn).



These model components are assumed to have variance, correlation,

and means. The variance terms are included in the model by creating additional latent standardized variables (i.e., z-scores). Since the variables labeled "z-level" and "z-slope" are assigned unit variance (i.e., 1) the regression coefficients or single arrows are interpreted as standard deviations Dev(L) and Dev(S), and their covariance is the correlation Corr LS. The inclusion of mean or intercept terms is usually not necessary in a path model, but here it is needed. To do this we add a new variable labeled a "constant" and draw it as a triangle. This triangle is drawn in RAMpath by entering "input_model variable(l)=constant". The intercept terms are represented by arrows from this constant to any other variable in the picture. Here we include parameters for the mean level Mean L and mean slope Mean S. More details on this model can be found in the associated references. (See Figure [M.2.5])

These are only a few examples of structural equation models being used today. A variety of more interesting examples can be found in the articles and books listed in the reference section. Graphic displays are becoming more and more important as the model increases in complexity. We think RAMpath diagrams such as those above can help make path analysis diagrams more standard and more accurate and, hence, more effective. All of the diagrams presented in this manual were drawn automatically by the RAMpath program.

M.3: The algebraic principles of RAM theory

The simple path analysis displays of simultaneous equations help communicate the overall analysis plan and the numerical results. In fact, these graphics are directly related to the matrix algebra expressions. Some properties of these rules were initially presented by McArdle (1978, 1979, 1980; Horn & McArdle, 1980) and termed RAM graphics. A few novel graphic principles were used to simplify all traditional path analysis principles. At the same time these graphic principles allowed us to be consistent with (1) the associated matrix algebra, (2) a simple set of "tracing rules" for structural expectations, and (3) structural expectations within correlational, covariance, and/or cross-product scalings. The revised RAM diagrams include all model parameters in a consistent fashion. Thus, RAM theory allows us to make it easy to draw and understand the implications of a path diagram.

The previous diagrams are simply a re-expression of an explicit set of algebraic principles. We know that a wide variety of linear structural

equation models can be represented by a simple set of linear equations (for proof, see McArdle & McDonald, 1984). This general algebra for structural models is presented in Table [M.3.1]. RAM algebra requires only three specification matrices (**A**, **S** and **F**), but these matrices are mainly empty or sparse. This mathematical model makes it clear how all variables and parameters of a model are organized and how this model organization makes explicit predictions about the expected connections (**C**) among all variables.

Table [M.3.1]: RAM algebraic notation (from McArdle & McDonald, 1984)

We write any linear structural equation model as:

$$(1) \quad v = v A + u, \text{ and}$$

$$S = \Sigma \{ u u' \}, \text{ where}$$

v = a $(t \times t)$ vector of all variables,

A = a $(t \times t)$ asymmetric matrix of regression coefficients among the v ,

u = a $(t \times t)$ vector of residual variables, and

S = a $(t \times t)$ symmetric matrix of expected statistics for the u .

It follows immediately that the expected values

$$(2) \quad E = (I - A)^{-1} = \sum_{r=0}^{\infty} A^r = A^0 + A^1 + A^2 + \dots + A^r + \dots + A^{\infty}$$

$$C = \Sigma \{ v v' \} = E S E', \text{ where}$$

E = the $(t \times t)$ asymmetric matrix of *Effects* among all v variables, and

C = the $(t \times t)$ symmetric matrix of *Connections* or expected statistics among all variables v .

Finally, if we define

$$(3) \quad v = [g; h], \text{ and}$$

$$F = [I; O], \text{ then}$$

$$X = \Sigma \{ g g' \} = F C F', \text{ where}$$

m = the $(l \times p)$ vector of manifest variables,

I = the $(1 \times q)$ vector of latent variables,

F = the $(p \times q)$ binary *Filter* matrix separating g from h , and

X = the expected summary statistics among the manifest variables.

The RAM model assumes each variable is assigned into an ordered vector of variables v , and each regression coefficient is assigned to a location in asymmetric matrix A ; i.e., $v = A v + u$, or $v = u (I-A)^{-1}$. Each variable $v(i)$ *internal* to the model is aligned with a $u(i)$ variable *external* to the model. These external variables $u(i)$ are not drawn in the graphic but they represent the random or stochastic source variables and are always aligned in a vector u . Without further details this rearrangement of the standard vector model into the equivalent matrix model using the basic RAM equation has a direct relationship to the graphic representation of the model: *The non-zero coefficients in asymmetric matrix A are always drawn as one-headed arrows from the column variable to the row variable.*

Any linear structural equation specification also requires the definition of any additional non-zero expectations. Again following the matrix notation of McArdle & McDonald (1984) we can write these expectations as a second equation; i.e., $S = \Sigma \{g g^T\}$, where Σ is the usual expectation operator. This implies the specification of the non-zero model expectations into a symmetric matrix S where the row and column subscripts for any element $S(r,c)$ are interchangeable. This rearrangement of the vector model into the matrix model also has a direct relationship to the graphic representation of the model: *The non-zero coefficients in symmetric matrix S are always drawn as two-headed slings between the column variable and the row variable.*

In most empirical studies we will need to make a clear distinction between the manifest and latent variables. In order to separate variables a third equation is defined where fixed coefficient matrix F is defined to *filter* the manifest variables g out of the vector of all variables v ; i.e., $g = Fv$, or $g = F(vA+u)$. This final RAM matrix F also has a graphic interpretation: *The unit-valued coefficients in filter matrix F represent the mapping of one variable in the graph with one manifest variable in the data.*

A simple example of the matrix algebra required for the latent variable path model of Figure [P.3.1] is presented in Table [M.3.2].

Table [M.3.2]: The latent variable path model using RAM algebra (see Figure [P.3.1])

First we define all variables as:

$$\begin{aligned} (1) \quad t &= 7, \quad v = [v1 \ v2 \ v3 \ v4 \ v5 \ v6 \ v7] \\ p &= 6, \quad g = [v1 \ v2 \ v3 \ v4 \ v5 \ v6], \text{ and} \\ q &= 1, \quad h = [v7] \end{aligned}$$

where

$v1, v2, v3$ = the observed independent variables,

$v4, v5, v6$ = the observed dependent variables, and

$v7$ = the unobserved common factor of the dependent variables.

Second we define the $(t \times t)$ matrix of asymmetric relations as:

$$(2) \quad A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P1 \\ 0 & 0 & 0 & 0 & 0 & 0 & P2 \\ 0 & 0 & 0 & 0 & 0 & 0 & P3 \\ B1 & B2 & B3 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where

B = the latent variable regression coefficients, and

P = the factor pattern coefficients.

Third we define the $(t \times t)$ matrix of symmetric coefficients as:

$$(3) \quad S = \begin{bmatrix} W1 & & & & & & \\ R12 & W2 & & & & & \\ R13 & R23 & W3 & & & & \\ 0 & 0 & 0 & U1 & & & \\ 0 & 0 & 0 & 0 & U2 & & \\ 0 & 0 & 0 & 0 & 0 & U3 & \\ 0 & 0 & 0 & 0 & 0 & 0 & Q1 \end{bmatrix}$$

where the upper and lower triangles of this matrix are identical,

$W_{..}$ = independent variable variance terms,

$R_{..}$ = independent variable covariance terms, and

Q_j = latent variable disturbance terms.

Fourth we define the (p x t) filter matrix as

$$(4) F = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

where the value of one is placed at location $F(i, j)$ to represent the matching of a variable $m(i)$ from the manifest variable vector with a variable $v(j)$ from the total variable vector. (Notice the only column without a 1 is the seventh column for latent variable v_7).

Finally, the matrices **E**, **C**, and **X** can be created from these matrices.

The key feature of RAM algebra is the direct graphic translation, and this applies to other aspects of the algebra as well. In this way the RAM algebra represents the *necessary and sufficient* sub-components of these general algebraic models in an effective graphic fashion. More details are available in the references listed at the end of this manual.

M.4: The graphic basis of RAM theory

McArdle and his colleagues (see McArdle, 1980; Horn & McArdle, 1980; McArdle & Horn, 1990) organized a few simple graphic principles that reflected the algebra of all structural equation models. We now provide a detailed treatment of these graphic principles in Table[M.4.1] below. Here we follow the digraph theory of Harary, Norman & Cartwright (1965). In this work a set of *primitives* are used to define necessary graphic terms and a set of *axioms* are used to explain the relevance of these definitions for structural models. We combine features of these prior treatments in all of our graphic displays and in the formal graphic definitions listed below.

Table [M.4.1]: Graphic Specification Rules for Linear Structural Equation Models.

Primitive (1): A two-dimensional non-cartesian space is termed a *graph*.

Axiom (1): Some graphs may be used to represent aspects of a *linear structural equation model*.

Primitive (2): The graphic elements in set $v = [v(1) \dots v(T)]$ are termed *points* (also vertices or nodes). The number of points T is termed the *size* of the graph.

Axiom (2): The set v may be used to designate the T *variables* in a structural equation model.

Primitive (3): The graphic elements in set $P = [P(1) \dots P(Q)]$ connect pairs of points and are termed *lines* (also edges or links). The number of lines Q is termed the *order* of the graph. The two points that are linked by a line are said to be *incident* to that line and *adjacent* to one another.

Axiom (3): The set P may be used to designate *parameters* in a model representing relationships among pairs of variables.

Primitive (4): The set P may be subdivided into two exclusive subsets **A** and **S**, so $P = [A : S] = [A(1) \dots A(Q(a)) : S(1) \dots S(Q(s))]$ and so $Q = Q(a) + Q(s)$. Each of these elements of P may take on a numerical value termed a *width*.

Axiom (4): The subdivision of P may be used to separate *asymmetric* relations (also projection or regression relations) from *symmetric* relations (also cross-product, covariance, or correlational relations). Numerical values may be assigned to a maximum of $Q(a) = T \cdot T$ asymmetric relations and a maximum of $Q(s) = T \cdot (T+1)/2$ symmetric relations.

Primitive (5): The set v may be subdivided into two exclusive subsets **m** and **l**, so $v = [g : h] = [g(1) \dots g(T(g)) : h(1) \dots h(T(h))]$ and so $T = T(g) + T(h)$.

Axiom (5): The subdivision of v may be used to separate *given* variables (also observed or manifest variables) from *hidden* variables (also unobserved or latent variables).

The second part of our model is a graphic expression of the well known "tracing rules of path analysis" (after Wright, 1918, 1921), or as the "components of a correlation" (after Fisher, 1918). In any model we can define a set of rules for combining parameters from their associated algebraic expressions. These rules are formally stated in Table [M.4.2].

Table [M.4.2]: The Graphic Action Rules for Non-Linear Structural Expectations.

Primitive (6): There exists a set of actions (also movements or tracings) within a graph.

Axiom (6): The graphic actions can be used to represent the *nonlinear structure* of any model expectations.

Primitive (7): The set E contains *actions* which are one-way movements or tracings in a graph. Each element $E[x \dots w]$ is the action from any variable $v(w)$ to any variable $v(x)$ that passes through a one-way asymmetric sequence of $r > 0$ adjacent variables connected together by $r-1$ one-headed arrows $A(i, j)$. The

length of an action is defined as $r-1$ (and ≥ 0), the number of arrows it contains, and the width of an action is defined as the product of the width of the $r-1$ (and ≥ 0) directed arrows. Every action of length 0 is assigned a width of 1.

Axiom (7): Each action $E[x...w]$ represents an *effect* of variable $v(w)$ upon variable $v(x)$. Any path with r regression coefficients is labeled as an *effect of length r* , and the value of this effect is the product of the r coefficients it joins. It is traditional to term effects of length one as *direct effects* and effects of lengths greater than one as *indirect effects*.)

Primitive (8): The set **B** contains *bridges* which are sequentially defined two-way movements or tracings in the graph. The element $B[x...w:z...y]$ is the tracing from any initial variable $v(x)$ to any terminating variable $v(y)$ that passes through a symmetric sequence of:

- (1) *one backward action* $E[x...w]$ tracing backwards from $v(x)$ to $v(w)$ and connecting $r > 0$ variables,
- (2) *one and only one span* $S(w,z)$ connecting two mediating variables $v(w)$ and $v(z)$, and
- (3) *one forward action* $E[y...z]=E[z...y]$ traced from $v(z)$ to $v(y)$ connecting $q > 0$ variables.

The width of the bridge is defined as the product of the widths of the two actions and one span.

Axiom (8): Each bridge $B[x...w:z...y]$ is a single *component* in the decomposition of some correlation, covariance, or cross-product term $C(w,z)$. The value of this component is the product of the values of the two effects $E[x...w]$ and $E[z...y]$ and the one and only one cross-product term $S(w,z)$.

Primitive (9): The set **C** contains *connections* which are complete linkages between pairs of variables. The element $C(x,y)$ is the sum of all of the non-parallel bridges $B[x...w:z...y]$ between the two variables $v(x)$ and $v(y)$ taken over all combinations of variables T up to maximum length z (i.e., for $x=1$ to T , $y=1$ to T , $r=0$ to z , and $q=0$ to z). Bridges are *non-parallel* if they may be stated in a different notational sequence even though they may contain the same elements (i.e., $B[x...w:z...y] \neq B[x...z:w...y]$). The width of the connection is defined as the sum of the widths or values of all of its non-parallel bonds.

Axiom (9): Every connection represents a correlation, covariance, or cross-product or other symmetric coefficient between two variables $C(x,y)$, and the numerical value of each connection is the sum of all independent components.

Primitive (10): The set of connections **C** can be subdivided into three exclusive subsets $C = [C(gg) : C(gh) : C(hh)]$ representing subsets of connections among the two subsets of variables.

Axiom (10): The structural relations among all variables may be simply subdivided into those representing relations among the manifest variables, $C(gg)$, those among only latent variables, $X = C(hh)$, and those representing relations between the manifest and latent variables, $C(gh)=C(gh)$.

M.5: RAMpath diagrams versus traditional Path diagrams

These simple RAM distinctions reflect all of the *necessary and sufficient* graphic and algebraic rules for the specification of *any* linear structural equation model. However, there are a few differences between standard path analysis and RAMpath notation. Wright (1918, 1982) clearly stated the graphic distinction between regressions drawn as one-headed arrows and all correlations drawn as curved lines with two arrowheads. Also, the graphic separation of circles and squares was used by others (e.g., Hauser & Goldberger 1971, where the circles and squares were reversed; Joreskog, 1974). Our RAMpath graphs are different because these diagrams *represent the complete structural equation model*.

One difference was used in most of our earlier pictures -- we eliminated the error or disturbance variables in the picture. This does not mean we cannot draw diagrams with error variances included. To the contrary, we can draw virtually any consistent diagram with RAMpath. Furthermore, as long as all model variables and parameters are included in the diagram, the tracing rules will always be correct.

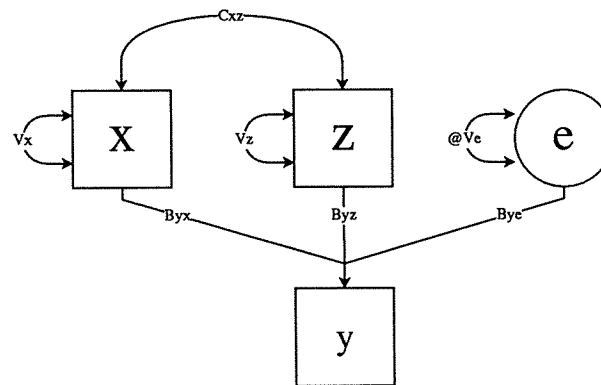


Figure [M.5.1]: Basic Regression Model (using RAM notation)

Figure [M.5.1] gives an example where the disturbance or error variable has been included in the model. This is a multiple regression model where observed variables x and z are assumed to affect observed variable y with independent error e . In this explicit drawing we can clearly see the traditional model assumption of the lack of a correlation between the error term and the independent predictors (x and z). In this model we can also assign a parameter to the arrow from e to y (i.e., the standard deviation of the unexplained variance). This makes a clear diagram useful for tracing purposes. Since, these assumptions are typical in all models, and a diagram with multiple variables becomes so much more complex, we usually do not draw the picture this way. Instead we prefer the compact graphic forms demonstrated earlier and we reserve this more typical approach for special cases. (See Figure [M.5.1])

A few nontraditional features of RAMpath diagrams are described in summary form in Table [M.5.1] below.

Table [M.5.1]: Some differences between RAM notation and Traditional Path Diagrams

- (R1) The Variable Inclusion Rule -- *All unobserved variables are labeled and ordered.* This simple addition implies all unobserved variables, including common factors, unique factors, and disturbance terms, are formally identical and are so treated.
- (R2) The Parameter Inclusion Rule -- *All model parameters are included in the diagram.* All model parameters representing projections (e.g., regressions or loadings) as well as all expectations (e.g., correlations and variances) among the stochastic variables are always drawn in the RAM graph. In traditional path diagrams the two-headed slings connecting the variable with itself are not included in the diagram. But these *self-slugs* usually represent the variance or sums-of-squares of a variable and these are valid parameters in any structural model. The parameters are needed for an accurate representation of the entire structural model.
- (R3) The Variable Exclusion Rule -- *If any variable $v(i)=v(j)$ has unit weight and no error, the $v(j)$ variable is redundant and need not be drawn.* This simple identity rule leads to a compact but equivalent algebraic expression, and the associated graphics are always smaller. This compact algebra and graphic creates useful diagrams and graphic heuristics emerge; i.e., We always interpret a sling as the residual of the impact of any arrows. These rules apply to models of any complexity.
- (R4) The Parameter Exclusion Rule -- *Any parameters which are linked by a latent variable with no sling attached can be summarized as the product of the values of the two parameters.* Symbolically, this simply means that if $v(x) \rightarrow v(z)=a$, $v(z) \rightarrow v(y)=b$, and $S(v(z) \rightarrow v(z))=0$, then $v(x) \rightarrow v(y)=ab=w$. This rule is mainly useful for reduced form calculations in advanced models.

Experienced path analysts will also notice that these graphic tracing principles are only slightly extended versions of the traditional path analysis tracing rules described by Wright (1921, 1985). For example, path analysts usually refer to the arrows of length one as both "paths" and "direct effects" and the sum of the arrows of greater length as "indirect effects" (e.g., see Alwin & Hauser, 1975; Fox, 1980). In RAM notation we simply separate all effect matrices and also add *effects of length zero*. These effects of various lengths are naturally explicit in both the graphic and the algebraic formulation of variables.

All RAMpath diagrams included here can be labeled consistent, or complete, because they include all algebraic parameters within the graphic display. This is not true of the traditional path analysis models (see Wright, 1934), the standard latent variable path models (Hauser & Goldberger, 1971), or the standard LISREL diagrams (see Joreskog & Sorbom, 1979, 1985). These traditional graphic displays do follow Sewell Wright's original correlational treatments where all variables were assumed to have unit variance. Unfortunately, these traditional simplifying assumptions create problems for latent variable models with possibly different scales (e.g., covariances, cross-products). Thus, the mathematical changes suggested by RAM theory are decidedly simple, but they are also nontrivial and have broader algebraic and interpretive results.

M.6: Statistical Principles

Statistical estimation is one of the cornerstones of structural equation modeling. These are the complex procedures for taking the summary statistics among the manifest variables and finding the best values for all unknown model parameters. As we mentioned earlier, one reason for increased interest in structural modeling is the availability of computer programs, such as the popular LISREL series (Joreskog & Sorbom, 1979, 1985) and other programs such as the less well-known COSAN programs (McDonald, 1985; Fraser & McDonald, 1989). These programs fit models to data, and calculate statistical estimates, but they do not draw pictures.

The current version of RAMpath does not provide parameter estimates -- Instead, RAMpath uses available parameter estimates to evaluate models. This RAMpath program is designed to be a useful adjunct to more standard estimation programs, like LISREL-7 (Joreskog & Sorbom, 1988), COSAN-II (Fraser & McDonald, 1978, 1988), PATHI

(Steiger, 1989), and CALIS (SAS Institute, 1990). Nevertheless, the displays provided by RAMpath can be usefully augmented with statistical information in several ways.

We use the fifth column of the internal model list to define whether or not a parameter is "fixed" or "free". This fifth column stands for the "Standard Error" of a parameter. Here we designate a fixed parameter by an ERROR=0, and we assume the parameter is free by an ERROR>0. We usually place the numerical value of the standard error in this location. For example, we can enter:

```
input_model
  arrow(1,2) = .5 free;
  /* assumes a standard error of 1 */
  arrow(1,3) = .5 fixed;
  /* assumes a standard error of 0 */
  arrow(1,4) = .5 ;
  /* assumes a standard error of current set_error */
```

We can also enter the free or fixed patterning using the symbols "@" for fixed and "?" for free. For example, we can enter:

```
input_model
  arrow(1,2) = .5 ? ;
  /* assuming a free parameter */
  arrow(1,3) = .5 @ ;
  /* assuming a fixed parameter */

/* or equivalently */
```

```
set_value=.5;
input_model_matrix
  arrows (4,4) =
    O ? @ O
    O O O O
    O O O O
    O O O O ;
```

These symbols can be printed on the diagram to display the location of fixed and free parameters (we draw the fixed @ automatically). We assume these symbols are mainly useful in large and complex models,

We also use the sixth column in a similar way to determine "equality constraints". Here we designate an EQUAL=0 if the parameter is not equal to any other parameter. But we add an EQUAL=N to define that

this parameter is equal to the Nth equality constraint. For example, we can enter:

```
/* suppose we wanted to make a few parameters equal to
one another */

input_model
  arrow(1,2) = .5 free 1;
  /* assumes equal to the first constraint */
  arrow(1,3) = .5 free 1;
  /* assumes equal to the first constraint */
  arrow(1,4) = .5 free 2;
  /* assumes equal to the second constraint */
  arrow(1,5) = .5 free 2;
  /* assumes equal to the second constraint */
```

At first this logic is somewhat complex, but it does allow users to enter such constraints at any time in the input. It also allows the definition of a wide variety of model constraints.

There are various ways to use and display these extra columns. Our inclusion of these columns now is relevant to further work on the display and estimation of an entire structural equation model. In later versions of RAMpath these statistical estimation techniques will be described in further detail.

M.7: Drawing Diagrams from SAS and LISREL

In our early work on RAM notation we tried to emphasize that these graphic displays were independent of any particular estimation algorithm (see Horn & McArdle, 1980). We used the early versions of LISREL and COSAN for model estimation. But these programs did not, and still do not, draw pictures. Thus an interface to RAMpath can be useful for any modeling program. We illustrate two such examples below.

The recent SAS program for structural modeling is called CALIS (see SAS, 1990). This procedure was written around the basic logic of the COSAN model and it includes most important structural estimation techniques. It also includes a direct interface to RAM notation, and this means drawing pictures from CALIS is relatively easy and automatic. The following SAS input file gives a simple example:

File [M.7.1]: An example of a SAS DATA step creating a RAM file (SAS2RAM.SAS)

- * This is an example of a SAS program which creates a valid RAMpath;
- * input file. This program uses an example from the SAS manual on;
- * PROC CALIS and shows how the model results can be entered and displayed.;
- * As long as the INRAM and OUTRAM conventions of CALIS are used.;
- * this file should work for any model fitted by PROC CALIS;

```
data spleen;
  input rosette nucleate;
  sqrtrose=sqrt(rosette);
  sqrtnucl=sqrt(nucleate);
  cards;
```

```
4 62
5 87
5 117
6 142
8 212
9 120
12 254
13 179
15 125
19 182
28 301
51 357
```

```
;
```

```
run;
```

```
proc calis data=spleen cov stderr method=ml ha=1
  ometh=congra outram=splram1;
  var sqrtrose sqrtnucl;
  ram 1 1 3 1,
      1 2 4 1,
      1 3 4 beta,
      2 1 1 .25,
      2 2 2 .25,
      2 3 3 v_dist,
      2 4 4 v_factnu;
  run;
```

```
data new;
  set splram1;
  retain SQUARES;
  FILE 'c:fromsas.ram';
  if _N_ = 1 THEN DO;
```

```
    SQUARES = _ROW_;
  END;
  if _TYPE_ = 'ESTIM' THEN DO;
    _MATNR_ = _MATNR_ - 1;
    IF _NAME_ = ' ' THEN DO;
      PUT 'input_model = " ' _MATNR_ _ROW_ _COL_
        _ESTIM_ _STDERR_ ' ' ';
      END;
    ELSE DO;
      PUT 'input_model = " ' _NAME_ * ' " ' _MATNR_
        _ROW_ _COL_ _ESTIM_ _STDERR_ ' ' ';
      END;
    END;
  if _TYPE_ = 'VARIABLE' AND _MATNR_ = 2 THEN DO;
    TEMP = 1;
    IF _COL_ > SQUARES THEN TEMP = 0;
    _MATNR_ = 0;
    PUT 'input_model = " ' _NAME_ * ' " ' _MATNR_ _COL_
      _COL_ TEMP ' ' ';
    END;
  RUN;
  DATA new1; set new;
    file 'c:fromsas.ram' MOD;
    put 'draw_model';
    stop;
  run;
```

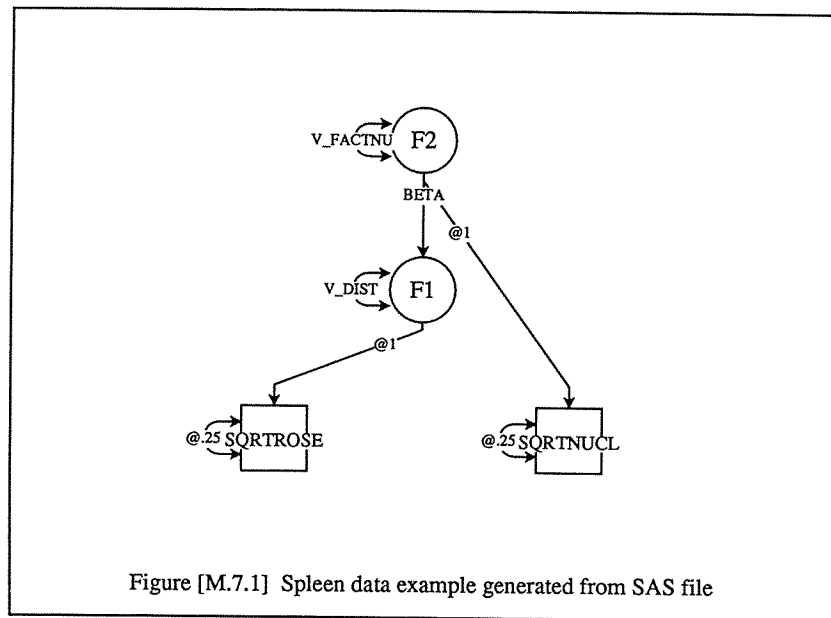
This file was run as is through the PC/SAS system and it produced the following output file:

File [M.7.2]: The resulting SAS output file (FROMSAS.RAM)

```
input_model = "SQRTROSE " 0 1 1 1 ;
input_model = "SQRTNUCL " 0 2 2 1 ;
input_model = "F1 " 0 3 3 0 ;
input_model = "F2 " 0 4 4 0 ;
input_model = " " 1 1 3 1 0 ;
input_model = " " 1 2 4 1 0 ;
input_model = "BETA " 1 3 4 0.390740157 0.0737996301 ;
input_model = " " 2 1 1 0.25 0 ;
input_model = " " 2 2 2 0.25 0 ;
input_model = "V_DIST " 2 3 3 0.3815281213 0.2734028958 ;
input_model = "V_FACTNU " 2 4 4 10.504584845 4.3905408777 ;
;
draw_model;
draw_bridges 1 2 postscript;
```

This file was then run through RAMpath to produce the following

picture (see Figure [M.7.1]).



This RAMpath diagram favorably compares with Figure 1.2 (p.14) of the CALIS manual, but our diagram was generated automatically. This interface between SAS/CALIS and RAMpath is fairly smooth and should be possible for models of even large complexity.

The most popular structural modeling program is probably LISREL-7 (Joreskog & Sorbom, 1988). It is also possible to use some optional features of LISREL-7 to help us draw a RAMpath diagram. The same example used above is now listed for a LISREL to RAMpath translation. First the LISREL input file is setup with a special OUTput card option:

File [M.7.3]: LISREL-7 Input file with save options

```
SPLEEN EXAMPLE
DA NI=2 NG=1 NO=12 MA=CM
RA
4 62
5 87
5 117
```

```
6 142
8 212
9 120
12 254
13 179
15 125
19 182
28 301
51 357
LA
  SQRTRSE SQRTRNUCL
SE
  1,2 /
MODEL NY=2 NE=4 BE=FU,FI PS=SY,FI LY=FU,FI TE=ZERO
LE
  SQRTRSE SQRTRNUCL FACTROSE FACTNUCL
FI LY 1 1 LY 2 2
VA 1 LY 1 1 LY 2 2
FI PS 1 1 PS 2 2
ST .25 PS 1 1 PS 2 2
FI BE 1 3 BE 2 4
ST 1 BE 1 3 BE 2 4
FR BE 3 4
ST .4 BE 3 4
FR PS 3 3 PS 4 4
ST .5 PS 3 3
ST 10.5 PS 4 4
OU LY=C:LIS2SAV.SAV BE=C:LIS2SAV.SAV PS=C:LIS2SAV.SAV NS
RS SE SO PT
```

We can use LISREL-7 to estimate the model parameters and produce a save file that looks like :

File [M.7.4]: LISREL-7 Save file output and
Revised LISREL output using RAMpath
notation

```
(8D13.6)          LY
.100D+01 .000D+00 .000D+00 .000D+00 .000D+00
.100D+01 .000D+00 .000D+00
(8D13.6)          BE
.000D+00 .000D+00 .100D+01 .000D+00 .000D+00
.000D+00 .000D+00 .100D+01 .000D+00 .000D+00
.000D+00 .392D+00 .000D+00 .000D+00 .000D+00
.000D+00
(8D13.6)          PS
.250D+00 .000D+00 .250D+00 .000D+00 .000D+00
.328D+00 .000D+00 .000D+00 .000D+00 .961D+01
(8D13.6)          EC
```

```
.597D-02 -.114D-02 .686D-01 -.183D-01 .174D-02
.177D+02
```

```
/* We then re-edit this file adding a few RAMpath matrix
commands */
```

```
set_title = "test run of lisrel-7 file - spleen data";
input_model_matrix
  filter (2,4) =
    .100D+01 .000D+00 .000D+00 .000D+00 .000D+00
    .100D+01 .000D+00 .000D+00 ;
input_model_matrix
  arrow (4,4) =
    .000D+00 .000D+00 .100D+01 .000D+00 .000D+00
    .000D+00 .000D+00 .100D+01 .000D+00 .000D+00
    .000D+00 .396D+00 .000D+00 .000D+00 .000D+00
    .000D+00;
input_model_matrix
  sling (4,!) =
    .250D+00 .000D+00 .250D+00 .000D+00 .000D+00
    .326D+00 .000D+00 .000D+00 .000D+00 .981D+01;
draw_model;
draw_bridges 1 2;
output_model;
output_model_matrix;
output_bridges;
```

After running this through the RAMpath program we have the same picture as listed above in Figure [M.7.1].

Using similar devices we can interface RAMpath with other structural modeling programs such as EzPath by James Steiger (1989) and COSAN from Fraser & McDonald (1989).

Part D. The Dictionary of RAMpath Commands

1. Introduction
2. Running RAMpath
3. Language Rules
4. Adding Comments
5. Aliasing Words for Numbers
6. DRAW_BRIDGES Command
7. DRAW_MODEL Command
8. INPUT_CLEAR Command
9. INPUT_MODEL Command
10. INPUT_MODEL_MATRIX Command
11. OUTPUT_BRIDGES Command
12. OUTPUT_MODEL Command
13. OUTPUT_MODEL_MATRIX Command
14. SAVE_MODEL Command
15. SET_COLOR Command
16. SET_EQUAL Command
17. SET_ERROR Command
18. SET_LABEL Command
19. SET_PRINT Command
20. SET_SHAPE Command
21. SET_SLING_ROTATE Command
22. SET_TITLE Command
23. SET_VALUE Command
24. SET_WIDTH Command
25. #DEFINE Command

Introduction

The first part of this dictionary describes some general rules used in running and writing RAMpath programs. The second part of this dictionary is an alphanumeric listing of all available commands in the current RAMpath computer program. Each command is listed together with its specific function, syntax and a few examples. The following commands are listed here:

Only a few of the available commands have been used up to this point. More examples of the use of this language are presented in the Examples section of the manual.

Running RAMpath

The RAMpath program is not interactive but runs in a variety of "batch" modes. In all cases RAMpath uses three files:

- (1) *.RAM - RAMpath input script file defining model.
- (2) *.OUT - Output from RAMpath including results of OUTPUT_ commands
- (3) *.SAV - RAMpath input script generated and output by SAVE_MODEL.
- (4) *.PS - Postscript file output by DRAW_ commands.

where * designates a filename provided by the user.

The simplest way to invoke the program is to respond to the MSDOS prompt with:

```
>RAMpath prefix
```

where prefix is the name of the input file *.RAM. In the above example the input line

```
>RAMpath TEST1
```

would read the TEST1.RAM file, and output the TEST1.OUT and TEST1.SAV files.

RAMpath Language Rules

Once inside the program, the user needs to use a specific programming language. One of the key features of RAMpath is the use of a flexible interpretive language for program input. (This language is called the MC interpreter by McArdle & Boker, 1986). These functions follow a mixture of C language conventions and simple commands. The general form of each command is:

```
command = # (and/or) "%" ;
```

where

- * the "command" is a specific word,
- * the pound sign # is a numerical value (or alias),
- * the quotes and percent sign "%" represent a quoted string, and
- * the command ends with a semicolon (;).

As a simple example we might enter

```
set_title = "This is a valid title line";  
set_print = 1;
```

Some generic language rules can be followed:

- * Commands: The program input is read as a series of commands.
- * Keyword: The first word used defines the command.
- * Numbers: All numbers read are used in sequence by the command.
- * Ending: All commands end in a semi-colon.
- * Recency: The most recent command takes precedence over any prior commands.
- * Defaults: All values are assumed to be "zero" unless input.
- * Comments: Use standard C-language "/" and "/" conventions (see below).
- * Parentheses: If Parentheses "(" and ")" are used they must balance.

Adding Comments

Function:

Comments may be placed inside the input text at any time using standard C-convention comment labels `/*` and `*/`. These are mainly used to document the input stream, but creative users may find comments useful in other ways. As a simple example we write

Syntax:

```
/* xxxxxx */
```

where xxxxx is any numbers or text.

Example:

```
/* This is a comment */  
  
/* So is  
this */  
  
/*  
This is a comment too  
*/
```

Note:

Comments do not affect the input program in any way. Comments cannot be nested inside one another. The `/*` and `*/` must balance.

ALIASING -- Changing words to Numbers

Function:

The language allows a selected set of keywords to be used as aliases or substitutes for numbers. These simply allow a more highly documented input and output without the need for quoted strings.

Syntax:

```
command alias; /* no quotes are used */
```

Example:

```
input_model  
  arrow(2,3) = .5 @ "3->2";
```

```
input_model_matrix  
  arrows (3,3) =  
    0 0 0  
    @ 0 0  
    ? 0 0;
```

```
/* In both cases above the word "arrow" is reinterpreted  
as the number 1 */
```

Notes:

The following keywords will be changed to numbers. Uppercase letters will be changed to lowercase letters by the program.

Keywords	Alias	Keywords	Alias	Keywords	Alias
arrow	1	asymmetric	1	filter	0
variable	3	sling	2	latent	0
circle	0	manifest	1	square	1
constant	3	triangle	3	symmetric	2
true	1	yes	1	false	0
invisible	-1	no	0	fixed	0
free	1	equal	2	normal	0
least	-2	less	-1	more	1
alot	2	debug	99	fixed	0
free	1	equal	2	start	3
F_	0	A_	1	S_	2
E_	3	C_	4	X_	5

DRAW_BRIDGES

Function:

DRAW_BRIDGES takes the input model and automatically generates several pictures representing the tracing rules of the path diagrams. These pictures are displayed on the CRT screen and they may be output to a printer.

Syntax:

```
DRAW_BRIDGES = ( #variable1 , #variable2 ) [postscript];
```

```
/*      where
```

```
#variable1 is the variable at the start of the bridge
```

```
#variable2 is the variable at the end of the bridge
```

```
[postscript] is an optional output to a postscript file
*/
```

Examples:

```
DRAW_BRIDGES = ( 5, 9 );
/* draws all bridges between variables 5 and 9 */
```

```
DRAW_BRIDGES = ( 4, 4 );
/* draws all bridges between variables 4 and 4 */
```

```
DRAW_BRIDGES = ( 4, 4 ) POSTSCRIPT;
/* draws all bridges between variables 4 and 4 and
saves the result in a postscript file for later printing
*/
```

Notes:

DRAW_BRIDGES will (1) draw the full path diagram with all variables and edges in dashed style; (2) highlight the two variables at the beginning and end of the specific connection; (3) highlight one bridge after another on the screen (requiring the user hit a <CR> to move to the next bridge); and (4) finish by highlighting all bridges in the connection. The use of the "postscript" option will automatically draw all screens to a postscript file named *.PS, and this file can be used to print this picture on any postscript-type laser printer.

DRAW_MODEL

Function:

DRAW_MODEL takes the current input model and automatically generates the path diagram. These pictures are displayed on the CRT screen and they may be output to a printer.

Syntax:

```
DRAW_MODEL [postscript];
```

```
/* draws a complete path diagram of the current model
with optional postscript output */
```

Examples:

```
draw_model;
/* draws the whole model to the screen */
```

```
draw_model postscript;
/* draws the whole model to the screen and *.PS file */
```

Notes:

DRAW_MODEL produces a diagram that is output to the CRT screen. This diagram may be printed immediately with the <PrtScr> Print Screen button (as long as GRAPHICS.SYS has been invoked). Otherwise, this diagram may be output as a postscript file.

INPUT_CLEAR

Function:

INPUT_CLEAR changes the internal model list to have zero parameters.

Syntax:

```
INPUT_CLEAR;
```

Example:

```
input_clear;
```

Notes:

This is useful for running many separate models in one execution of the program. This command can be used at any time and it does not produce any output.

INPUT_MODEL

Function:

INPUT_MODEL is the primary way that a model is input to RAMpath. This command requires each parameter in a model to be entered one at a time in a structured fashion. The INPUT_MODEL is a structure which represents a single model parameter and has values which represent characteristics of the parameter. The collection of all model parameters input in this way is called a "Model List".

Syntax:

```
/* This command can follow a simple word-number  
combination as follows */
```

```
INPUT_MODEL #matrix #row #column [#value [#error] ]  
["label"];
```

where

#matrix = The number or keyword of the matrix of the parameter;
The keywords "arrow", "sling", "filter", or "variable" are valid keywords here.

#row = The row number of the parameter.
This is not used in "variable" structure;

#column = The column number of the parameter;

#value = The optional input of a numerical value of the parameter;
The keywords "manifest", "latent", "constant", and "invisible" are useful with the "variable" command.

#error = The optional input of the standard error of the parameter
The keywords "fixed" and "free" as well as "@" and "?" may be used as aliases in this command.

"label" = A quoted string which serves as the label of the parameter;

Examples:

Some examples using word-number commands follow:

```

input_model

  arrow(2,1) = .5;
  /* the arrow from 1 to 2 is given a value of .5 */

  arrow(2,1) = .5 free ;
  /* the arrow from 1 to 2 is .5 and free */

  arrow(2,1) = .5 ? ;
  /* the arrow from 1 to 2 is .5 and free */

  arrow(2,1) = .5 "Regression";
  /* the arrow from 1 to 2 is given a value of .5 and
  labeled "Regression" */

  arrow(3,4) = 1 fixed ;
  /* the arrow from 4 to 3 is fixed at 1 */

  arrow(3,4) = 1 @ ;
  /* the arrow from 4 to 3 is fixed at 1 */

  arrow(4,4) = .25 @ ;
  /* the arrow from 4 to itself is fixed at .25 */

/* Note: These commands are order dependent, so the
numbers and the order they are entered are critically
important. */

```

```

input_model

  sling(1,1) = .75;
  /* the sling between 1 and itself is given a value of
  .75 */

  sling(2,1) = .5 free ;
  /* the sling between 1 to 2 is .5 and free */

  sling(3,3) = 1 fixed ;
  /* the sling on variable 3 is fixed at 1 */

  sling(1,2) = 1 fixed ;
  /* the sling between variable 1 and 2 is fixed at 1 */

  sling(2,1) = .5 "Correlation" ;
  /* the sling between 1 to 2 is .5 and labeled
  "Correlation" */

```

```

input_model

  filter(1,1)=manifest;
  /* variable number 1 is manifest */

  filter(2,2)=manifest;
  /* variable number 2 is manifest */

  filter(1,3)=latent;
  /* variable number 3 is latent */

  filter(2,1)=manifest;
  /* variable number 2 in the data is variable number 1
  in the model and it is manifest */

  filter(1,1)=manifest "X1";
  /* variable number 1 is manifest and labeled "X1" */

/* Note: This use of the variable subcommand is an easy
way to create a filter matrix */

```

```

input_model

  variable(1)=manifest;
  /* variable number 1 is manifest */

  variable(2)=manifest;
  /* variable number 2 is manifest */

  variable(3)=latent;
  /* variable number 3 is latent */

  variable(4)=constant;
  /* variable number 4 is constant */

  variable(5)=invisible;
  /* variable number 5 is included but not drawn */

  variable(1)=manifest "X1";
  /* variable number 1 is manifest and labeled "X1" */

```

A more general approach can be described using only numerical conventions.

General Syntax:

```

INPUT_MODEL #matrix #row #column #value #error [##]
["label"] ;

```

/* where the first five numbers are a required list of numbers or words, ## is an optional list of additional numbers or words, and "label" is an optional label for the parameter which will be printed */

The following columns are defined for both input and output:

Column 1 = MATRIX Number of specific parameter (Number of Arrowheads):

This number can be one of three integers:

- 0 = No headed (F_ matrix element for Filter Relation)
- 1 = One headed (A_ matrix element for Asymmetric Relation)
- 2 = Two headed (S_ matrix element for Symmetric Relation)

(Note:

- 3 = "variable" statement (F matrix element calculated))

Column 2 = ROW Number of Variable in specific matrix (Output variable):

This number can be any integer between 1 and M (the number of variables). The assignment of row or column variable does not matter in the Symmetric relations (S-matrix). (This column is not used by the "variable" alias")

Column 3 = COLUMN Number of Variable in specific matrix (Input variable):

Column 4 = VALUE is the Numerical Value of Parameter:

This number is usually given in decimal notation and designates the "starting value" on input and the "estimated value" on output.

The remaining columns are usually optional.

Column 5 (optional) = ERROR is the Numerical Value of Standard Error.

This number is usually given in decimal notation and designates the "free or fixed status" on input (i.e., if a value of zero is assigned the parameter is fixed) On output, this value can be used to form the estimated $T = (\text{Parameter} / (\text{Standard Error}))$ value.

Column 6 (optional) = PATTERN = Number of Any Equality Constraint:

In most cases, the parameter is not forced to be equal to any other so this value is zero. However, if the parameter is forced to be equal to one or more other parameters this value is an integer which is given the same value for all set equal to one another (e.g., 1, 2, 3, etc.)

Column 7 (optional) = SCALED parameter value on output:

In some cases the parameters are interpreted in standardized form where all variables have unit variance. This feature is only used on output.

Column 8 (optional) = SHAPE of Drawing Character

This number represents the shape of the node or edge to be drawn.

If the list element is a variable, the following shapes are used:

- 0 = Invisible variable. Not drawn.
- 1 = Variable is drawn as a circle.
- 2 = Variable is drawn as a square.
- 3 = Variable is drawn as a triangle.

If the list element is a parameter, the following shapes are used:

- 0 = Invisible parameter. Not drawn.
- 1 = Single headed arrow.
- 2 = Double headed sling.

Column 9 (optional) = COLOR Drawing Line Style

The color is shown as color on the graphics screen and as a grayscale when printed as postscript.

Column 10 (optional) = WIDTH of Drawing Line Thickness

The width of the line drawn is represented by this number. Legal values are between 0 and 16. A value of 0 for the line thickness will cause the line thickness to be proportional to the value of the parameter multiplied by the error value.

Examples:

```
input_model
1,1,2,.5,1,0,0,1,1,3 "arrow from 2 to 1";
2,1,2,.5,1,0,0,1,1,3 "sling from 2 to 1";
```

INPUT_MODEL_MATRIX

Function:

INPUT_MODEL_MATRIX is one way that a model is input to RAMpath. This command requires each parameter in a model to be entered one at a time in a structured fashion. The INPUT_MODEL_MATRIX is a structure which represents an entire set of model parameters and has values which represent characteristics of the parameter. Typically three matrices are input, A, S, and F. These matrices contain elements which have numerical values and a specific pattern (i.e., fixed, free, or equal). The program accepts these parameters as a starting point and develops a model list from the non-zero elements.

*Syntax:

```
INPUT_MODEL_MATRIX #matrix #rows #columns = ##values;
```

```
/*
```

The #matrix name may be either "filter", "arrow", "sling", or "variable";

The #row gives the number of rows to input;

The #column gives the number of columns to input;

The number of values must be ##elements = #rows * #columns unless the "!" operator is used.

The "!" designates a matrix order which is lower triangular;

The "@" designates a parameter which is fixed at 1 or a SET_VALUE <> 0;

The "?" designates a free parameter started at 1 or a SET_VALUE <> 0;

```
*/
```

Examples:

```
input_model_matrix;
  arrow(3,3) =
    0 0 1      /* the arrow(1,3) with VALUE=1 */
    0 0 .5     /* the arrow(2,3) with VALUE=.5 */
    0 0 0;
```

```
input_model_matrix;
  arrow(3,3) =
    0 0 @      /* arrow(1,3) is fixed at 1 */
    0 0 ?      /* arrow(2,3) is free but starts at 1 */
    0 0 0;
```

```
input_model_matrix
  slings(3,3) =
    /* the sling matrix is square and symmetric */
    1 0 0
    0 1 .5
    .5 0 1;
```

```
input_model_matrix
  slings(3,!) =
    /* the sling matrix is a lower triangle */
    1
    0 1
    .5 0 1;
```

```
input_model_matrix
  filter
    1 0 0
    0 1 0;
```

Notes:

This command reads three initial numbers -- the matrix number, the row size, and the column size. The row and column sizes are used to determine the number of the values to be read in next, and the matrix name is used as the final storage location. The number of values input must be equal to the product of the row and column values or an error will be reported. These matrices are not saved but the non-zero elements are turned into elements of the internal model list.

OUTPUT_BRIDGES

Function:

OUTPUT_BRIDGES takes the current input model and provides a tabular listing of the full matrix expansion to the output file. This listing uses a text notation with arrowheads to designate the specific tracing rules and their associated values.

Syntax:

```
OUTPUT_BRIDGES #variable1 #variable2;

/*
where
    #variable1 is the variable at the start of the bridge
    #variable2 is the variable at the end of the bridge
*/
```

Examples:

```
output_bridges 1 2;
/* output all bridges between variables 1 and 2 */

output_bridges 1 1;
/* output all bridges between variables 1 and 1 */
```

Notes:

OUTPUT_BRIDGES uses arrowhead notation (1->2<->2->3) and produces a detailed listing that is output to the *.OUT file.

OUTPUT_MODEL

Function:

OUTPUT_MODEL takes the current input model and provides a tabular listing to the output file. This listing is identical to the numerical information in the internal model list and can be useful for checking models. This command can be used at any time.

Syntax:

```
OUTPUT_MODEL [matrix_type] [ text | numeric ] ;

/* where matrix_type is arrow, sling, or filter to give
a listing of only one part of the current model list.
if the option 'text' is supplied, the matrix column will
be printed using the matrix_type alias, ie. filter,
arrow, sling.
if the option 'numeric' is supplied, the matrix column
will be printed using the matrix number, ie. 0, 1, 2.
*/
```

Examples:

```
output_model;
output_model arrows;
output_model slings;
output_model filters;
```

Notes:

OUTPUT_MODEL produces a list of the current model that is output to the *.OUT file.

OUTPUT_MODEL_MATRIX

Function:

OUTPUT_MODEL_MATRIX takes the current input model and provides a matrix listing to the output file. The initial part of this listing includes the **A** matrix of arrows, the **S** matrix of slings, and the **F** matrix of filters. OUTPUT_MODEL_MATRIX also does a numerical calculation from any model list. The second part of this listing includes the Effect matrix inverse expansion $E = (I - A)^{-1}$, the **C** matrix of all connections, and the **X** matrix of model expectations.

Syntax:

```
OUTPUT_MODEL_MATRIX #matrix;
```

```
/* where #matrix_number is 0 to 5 to give a listing of  
only one part of the current model list */
```

Examples:

```
output_model_matrix;  
output_model_matrix A_  
output_model_matrix S_  
output_model_matrix F_;
```

Notes:

OUTPUT_MODEL_MATRIX produces a listing that is output to the *.OUT file.

SAVE_MODEL

Function:

SAVE_MODEL takes the current input model and provides a machine readable listing to a specified output file.

Syntax:

```
SAVE_MODEL ["save_file_name"] [ text | numeric ] ;
```

The "save_file_name" is an optional name of the textfile to which the model should be saved. If no "save_file_name" is supplied, the textfile will have the same name as the input script file with the extension .SAV.

If the option 'text' is supplied, the model will be saved using the aliases that make the list more human readable.

If the option 'numeric' is supplied, the model will be saved using the numeric representation of all columns. This is useful for translating a RAMpath model into a format that can be read by other programs such as SAS/CALIS.

The number of columns that are printed is dependent on the value of the SET_PRINT variable. If SET_PRINT = normal, then the columns up to and including the pattern column will be printed. If SET_PRINT = less, then columns up to and including the error column will be printed. If SET_PRINT = more, then all columns will be printed including the optional graphics only columns.

Examples:

```
SAVE_MODEL;
```

```
SAVE_MODEL "extra.sav";
```

Notes:

SAVE_MODEL produces a listing that is readable as input to RAMpath but it does not use word number aliases. Thus, commands such as INPUT_MODEL are listed with full numerical representation (i.e., the input "arrows" are replaced by "1", etc.)

SET_COLOR

Function:

SET_COLOR changes the default color to be equal to the input value. This is useful for highlighting variables or parameters. This command can be used at any time and it does alter the DRAW_MODEL output.

Syntax:

```
SET_COLOR = #draw;

/*
where #color_number may be
    0 = invisible
    1 = white
    2 = gray
    3 = yellow
    4 = red
*/
```

Example

```
set_color = 2;

input_model
    arrow(1,2)=.5;
/* the arrow from 2 to 1 will have a color of 2 */
arrow(3,2)=.1;
/* the arrow from 2 to 1 will have a color of 2 */
```

Notes:

SET_EQUAL

Function:

SET_EQUAL allows the user to change the default of model list at column six. If a sixth parameter is not explicitly entered into the model this SET_EQUAL value will be used. This can also be done by using the INPUT_MODEL with a sixth column, but SET_EQUAL may be an easier way to accomplish the same goal. This command can be used at any time and it does not directly alter the model list.

In most cases, the parameter is not forced to be equal to any other so this value is zero. However, if the parameter is forced to be equal to one or more other parameters this value is an integer which is given the same value for all set equal to one another (e.g., 1, 2, 3, etc.)

Syntax:

```
SET_EQUAL #equal;
```

Examples:

```
input_model
    arrow(1,2);

set_equal = 1;
/* 1 implies equality constraint to parameter #1 */
input_model
    arrow(1,3);
    arrow(1,4);
```

Notes:

SET_ERROR

Function:

SET_ERROR allows the user to change the default for the model list at column five. If a fifth parameter is not explicitly entered into the model this SET_ERROR value will be used. This can also be done by using the INPUT_MODEL with a fifth column, but SET_ERROR may be an easier way to accomplish the same goal. This command can be used at any time and it does not directly alter the model list.

Syntax:

```
SET_ERROR #error;
```

Examples:

```
set_error = fixed;    /* 0 implies a fixed parameter */
input_model
  arrow(1,2);
  sling(2,2);

set_error = free;      /* 1 implies a free parameter */
input_model
  arrow(1,3);
  sling(2,3);
  sling(3,3);

/* The first two parameters are fixed and the last three
are free */
```

Notes:

When RAMpath begins, SET_ERROR defaults to free. SET_ERROR does not generate any parameters. The aliases for fixed parameters are the number 0, "fixed", or "0";. The aliases for free parameters are the number 1, "free", or "1";.

SET_LABEL

Function:

SET_LABEL changes the default label to be equal to the input string. This is useful for labeling a set of parameters or variables. This command can be used at any time and it does alter the DRAW_MODEL output.

Syntax:

```
SET_LABEL = "label";
```

Example

```
set_label = "factor loadings";
input_model
  arrow(1,5);
  arrow(2,5);
  arrow(3,5);
  arrow(4,5);

/* all of these parameters will have a label called
"factor loadings" */
```

Notes:

Both variables and parameters may be labeled. If labels are not used, the numeric values are printed on the diagrams. The maximum length of any label is 20 characters, but in practice labels longer than 5 or 6 characters tend to be hard to read on the diagram.

SET_PRINT

Function:

SET_PRINT controls the amount of output produced by RAMpath. Increasing values (or aliases) produce more output in the fashion defined below. This command does not alter the calculations in any way.

Syntax:

```
SET_PRINT = #level;

/*
where
  -1 = "less" output;
   0 = "normal" output;
   1 = "more" output;
   2 = "alot" of output;
  99 = "debug" with voluminous output;
*/
```

Examples:

```
set_print = 1;
set_print = more;
set_print = less;
```

Notes:

This command may be most useful for debugging.

SET_SHAPE

Function:

SET_SHAPE changes the default SHAPE to the current input value. This is useful for changing the shape of the output variables or parameters. This command can be used at any time and it does alter the DRAW_MODEL output.

Syntax:

```
SET_SHAPE = #shape;

/*
where #shape may be
      0 = invisible
      1 = circles
      2 = squares
      3 = triangles
*/
```

Example

```
set_shape = 3;
input_model
  variable(1);
/* the variable 1 will have a SHAPE of 3, a triangle */
  variable(2);
/* the variable 2 will have a SHAPE of 3, a triangle */
```

Notes:

It is often useful to use SET_SHAPE in a different way for different kinds of variables, and also in a different way for variables and parameters.

SET_SLING_ROTATE

Function:

SET_SLING_ROTATE changes the DRAW_MODEL output so self-slings at the top and bottom of the graphic point up and down. The usual default is for all self-slings to point in from the left-hand side. This command can be used at any time and it does not produce any output.

Syntax:

```
SET_SLING_ROTATE = #true;
```

Example

```
draw_model;  
  
set_sling_rotate = true;  
  
draw_model;  
/* to try out the new look */
```

Notes:

Users need to experiment with this command to see if it alters their diagrams.

SET_TITLE

Function:

SET_TITLE is used to document the output. Text strings are entered and these are printed before the output that follows. These title lines can be changed as the program is running, and more than one title line may be used.

Syntax:

```
SET_TITLE = [ #line ] "title";
```

Examples:

```
set_title = "Figure 1: first try of model 1";  
  
set_title 2 = "with a second title line";  
  
set_title 3 = "with a third title line";
```

Notes:

SET_TITLE is a useful way to document both input and output. SET_TITLE is usually the first command in a RAMpath input script.

SET_VALUE

Function:

SET_VALUE allows the user to change the default for the model list at column four. If a fourth parameter is not explicitly entered into the model this SET_VALUE number will be used. This can also be done by using the INPUT_MODEL with a fourth column, but SET_VALUE may be an easier way to accomplish the same goal, especially for a large number of parameters. This command can be used at any time and it does not directly alter the model list.

Syntax:

```
SET_VALUE = #value;
```

Examples:

```
set_value = 1;
input_model
    arrow(1,2);
    sling(2,2);
```

```
set_value = .5;
input_model
    arrow(1,3);
    sling(2,3);
    sling(3,3);
```

```
/* so the first two parameters have VALUE=1 and the last
three have VALUE=.5 */
```

Notes:

When RAMpath starts, the default SET_VALUE is 1. SET_VALUE does not generate any parameters. No aliases are used by SET_VALUE.

SET_WIDTH

Function:

SET_WIDTH changes the default WIDTH to be equal to the input value. This is useful for highlighting variables or parameters. This command can be used at any time and it does alter the DRAW_MODEL output.

Syntax:

```
SET_WIDTH = #width;
```

```
/*
where #width may be
    0 = the width will be proportional to the value.
    1-16 = fixes the width. 1 is narrow, 8 is wide.
*/
```

Example

```
set_width = 7;
input_model
    arrow(1,2)=.5;
/* the arrow from 2 to 1 will have a width of 7 */
    arrow(3,2)=.1;
/* the arrow from 2 to 1 will have a width of 7 */
```

Notes:

It is often useful to use SET_WIDTH in a different way for different kinds of variables, and also in a different way for variables and parameters. It can be very useful to SET_WIDTH=0 before entering the slings and arrows. In this way the width of the slings and arrows will give information about their relative importance to the model.

#DEFINE

Function:

#DEFINE is used to change the name of variables to numbers for easy reference. This command provides a temporary alias for a particular number. This command will be retained for use by the input program, but it will not affect the output.

Syntax:

```
#DEFINE #variable = "label";
```

Example

```
#define 1 = "Motive";  
#define 2 = "Intellect";  
#define 3 = "Emotion";
```

```
input_model  
    arrow ( Motive, Intellect ) = .5;  
    sling ( Motive, Intellect ) = .75;  
    sling ( Intellect, Intellect ) = 1;  
    sling ( Emotion, Emotion ) = 1;
```

```
draw_model;
```

Notes:

Spelling of the variable names is essential. This command is not essential to the running of the program, but it may be useful for documentation or for teaching.

Part R: Selected RAMpath References

- Alwin, D.F. & Hauser, R.M. (1975). The decomposition of effects in path analysis. *American Sociological Review*, 40, 37-47.
- Alwin, D.F. & Tessler, R.C. (1975). Causal models, unobserved variables, and experimental data. *American Journal of Sociology*, 80, 58-86.
- Bagozzi, R.P. (1980). *Causal Models in Marketing*. New York: Wiley.
- Bagozzi, R.P., Fornell, C. & Larcker, D.F. (1981). Canonical correlation analysis as a special case of a structural relations model. *Multivariate Behavioral Research*, 16, 437-454.
- Bentler, P.M. & Weeks, D.G. (1985). Some comments on structural equation models. *Brit. Jour. Math. Stat. Psych.*, 38, 120-121.
- Berry, W.D. (1984). *Nonrecursive Causal Models*. Beverly Hills, CA: SAGE Press.
- Blalock, H.M. (1985). *Causal Models in the Social Sciences (2nd Ed.)*. New York: Aldine Pub.
- Blalock, H.M. (1985). *Causal Models in Panel and Experimental Designs*. New York: Aldine Pub.
- Bock, R.D. & Bargmann, R.E. (1966). Analysis of covariance structures. *Psychometrika*, 31, 507-533.
- Boker, S.M. PRINTRIX (1986). Typesetting and compositing software. Denver: Data Transforms, Inc.
- Boker, S.M. FONTRIX (1983). Font creation and virtual graphics software. Denver: Data Transforms, Inc.
- Boker, S.M. GRAPHTRIX (1980). Dot matrix graphics printing software. Denver: Data Transforms, Inc.
- Bollen, K.A. (1989). *Structural Equations with Latent Variables*. New York: Wiley.
- Browne, M. (1982). Covariance structures. In D.M. Hawkins (Ed.), *Topics in Applied Multivariate Analysis*. Cambridge: Cambridge University Press.
- Cattell, R.B. (1978). *The Scientific Use of Factor Analysis in Behavioral and Life Sciences*. New York: Plenum Press.
- Cattell, R.B. (1966). Higher-order factor structures and reticular versus hierarchical formula for their interpretation. In C. Banks & P.L. Broadhurst (Eds.), *Studies in Psychology*. London: University of London Press.
- Carey, G. (1986). A general multivariate approach to linear modeling in human genetics. *Am. J. Hum. Genet.*, 39, 775-786.
- Cudeck, R. (1984). Review of "Confirmatory factor analysis and covariance structure models" by J.S. Long. *Applied Psychological Measurement*, 8, 365-367.
- Duff, I.S. & Reid, J.K. (1979). Some design features of a sparse matrix code. *ACM Transactions on Mathematical Software*, 5 (1), 18-35.

- Duncan, O.D. (1975). *Introduction to Structural Equation Models*. New York: Academic Press.
- Duncan, O.D. (1966). Path analysis -- sociological examples. *American Journal of Sociology*, 72, 1-16.
- Englehart, M.D. (1936). The technique of path coefficients. *Psychometrika*, 1 (4), 287-293.
- Everitt, B.S. & Dunn, G. (1983). *Advanced Methods of Data Exploration and Modeling*. London: Heinemann.
- Fraser, C. (1979). *COSAN Users' Guide*, Ontario Institute for Studies in Education, Toronto (Note: or Center for Behavioral Studies, University of New England, Armidale, N.S.W., Australia).
- Fraser, C. & McDonald, R.P. (1988). COSAN: Covariance structure analysis. *Multivariate Behavioral Research*, 23, 263-265.
- Fornell, C. (1982) (Ed.), *A Second Generation of Multivariate Analysis, Volumes 1 & 2*. New York: Praeger.
- Fox, J. (1984). *Linear Statistical Models & Related Methods: With Applications to Social Research*. New York: Wiley.
- Fox, J. (1980). Effect analysis in structural equation models. *Sociological methods & research*, 9 (1), 3-28.
- Fraser, C. & McDonald, R.P. (1988). COSAN: Covariance Structure Analysis. *Multivariate Behavioral Research*, 23 (1), 263-265.
- Fulker, D.W. & DeFries, J.C. (1983). Genetic and enviornmental transmission in the olorado Adoption Project: Path analysis. *Brit. Jour. Math. Statis. Psych.*, 36, 175-188.
- Glymour, C. Scheines, R., Spirtes, P. & Kelly, K. (1987). *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling*. New York: Academic Press.
- Graff, J. & Schmidt, P. (1982). A general model for the decomposition of effects. In Joreskog, K.G. & Wold, H.O. (Eds.), *Systems under indirect observation: Causality, structure, prediction*. Amsterdam: North Holland. 131-148.
- Hanushek, E.A. & Jackson, J.E. (1977). *Statistical Methods for Social Scientists*. New York: Academic Press.
- Harary, F., Norman, R.Z. & Cartwright, D. (1965). *Structural models: An introduction to the theory of directed graphs*. New York: Wiley.
- Hauser, R.M. & Goldberger, A.S. (1971). The treatment of unobservables in path analysis. In Costner, H.L. (Ed.), *Sociological methodology 1971*. San Francisco: Jossey-Bass, 81-117.
- Hayduk, L.A. (1987). *Structural Equation Modeling with LISREL*. Baltimore: Johns Hopkins University Press.

- Hodapp, V. & Wermuth, N. (1983). Decomposable models: A new look at interdependence and dependence structures in psychological research. *Multivariate Behavioral Research*, 18, 361-390.
- Hodge, R.M. & Trieman, D.J. (1968). Social participation and social status. *American Sociological Review*, 722-740. New York: Wiley.
- Horn, J.L. & Engstrom, R. (1979). Cattell's scree test in relation to Bartlett's chi-square test and other observations on the number of factors problem. *Multivariate Behavioral Research*, 14, 283-300.
- Horn, J.L. & McArdle, J.J. (1980). Perspectives on mathematical and statistical model building (MASMOB) in aging research. In L. W. Poon (Ed.), *Aging in the 1980s: Contemporary Perspectives*. Washington, D. C.: American Psychological Association.
- Jackson, D.J. & Borgatta, E.F. (1981) (Eds.). *Factor Analysis and Measurement in Sociological Research*. Beverly Hills: Sage.
- James, L.R., Mulaik, S.A. & Brett, J.M. (1982). *Causal Analysis: Assumptions, Models and Data*. Beverly Hills, CA: SAGE Press.
- Joreskog, K.G. (1982). Analysis of covariance structures (with comments). In C. Fornell (Ed.), *A second generation of multivariate analysis, Volume 1*. New York: Praeger. 200-266. (Also in 1981, *The Scandinavian Journal of Statistics*, 8, 65-92).
- Joreskog, K.G. (1974). Analyzing psychological data for structural analysis of covariance matrices. In R.C. Atkinson, D.H. Krantz, R.D. Luce & P. Suppes (Eds.), *Contemporary developments in mathematical psychology, Volume II*. San Francisco: W.H. Freeman, 1-56.
- Joreskog, K.G. (1973). A general method for estimating a linear structural equation system. In A.S. Goldberger & O.D. Duncan (Eds.), *Structural equation models in the social sciences*. New York: Seminar Press.
- Joreskog, K.G. (1973b). Analysis of covariance structures with means. In A.S. Goldberger & O.D. Duncan (Eds.), *Structural equation models in the social sciences*. New York: Seminar Press.
- Joreskog, K.G. & Goldberger, A.S. (1975). Estimation of a model with multiple indicators and multiple causes of a single latent variable. *Jour. Amer. Stat. Assoc.*, 70, 631-639.
- Joreskog, K.G. & Sorbom, D. (1985). *LISREL-VI Program Manual*. Chicago: International Educational Services.
- Joreskog, K.G. & Sorbom, D. (1979). *Advances in Factor Analysis and Structural Equation Models*, (J. Magidson, Ed.). Cambridge, Mass.: Abt Books.
- Joreskog, K.G. & Sorbom, D. (1977). Statistical models and methods for the analysis of longitudinal data. In D. V. Aigner & A. S. Goldberger (Eds.), *Latent variables in socioeconomic models*. Amsterdam: North Holland.
- Joreskog, K.G. & Wold, H. (1982). *Systems Under Indirect Observation, Parts I & II*. Amsterdam: North Holland.

- Judd, C.M. & Kenny, D.A. (1979). *Estimating the Effects of Social Interventions*. Cambridge: Cambridge University Press.
- Kessler, R.C. & Greenberg, D.F. (1981). *Linear Panel Analysis: Models of Quantitative Change*. New York: Academic Press.
- Kiiviri, H. (1982). Comments on "Analysis of covariance structures" In C. Fornell (Ed.), *A second generation of multivariate analysis, Volume 1*. New York: Praeger, 247-250. (Also in 1981, *The Scandinavian Journal of Statistics*, 8, 83-89).
- Kiiviri, H. & Speed, T.P. (1982). Structural analysis of multivariate data: A review. In S. Leinhardt (Ed.), *Sociological methodology 1982*. San Francisco: Jossey Bass, 209-289.
- Lawley, D.N. & Maxwell, A.E. (1971). *Factor Analysis as a Statistical Method*. New York: American Elsevier Pub.
- Li, C.C. (1975). *Path analysis - a primer*. Pacific Grove, A: Boxwood Press.
- Li, C.C. (1969). Fisher, Wright, and path coefficients. *Biometrics*, 24, 471-483.
- Loehlin, J.C. (1987). *Latent Variable Models: An Introduction to Factor, Path, and Structural Analysis*. Hillsdale, N.J.: Erlbaum.
- Lohmoeller, J-B. (1984). *LVPLS 1.6 Program Manual: Latent Variable Path Analysis with Partial Least Squares Estimation*. Central Archive for Empirical Social Science, University of Cologne.
- Marsden, P.V. (1981) (Ed.), *Linear Models in Social Research*. Beverly Hills, CA: SAGE Press.
- McArdle, J.J. (1989). A structural modeling experiment with multiple growth functions. In R. Kanfer, P.L. Ackerman & R. Cudeck (Eds.), *Abilities, Motivation, and Methodology*. Hillsdale, NJ: Lawrence Erlbaum Assoc., Pub., pps. 71-117.
- McArdle, J.J. (1988). Dynamic and structural equation modeling with repeated measures data. In J. R. Nesselroade & R. B. Cattell (Eds.), *Handbook of Multivariate Experimental Psychology, Volume II*, pg 561-614. New York: Plenum Press. (In press).
- McArdle, J.J. (1986). Latent growth within behavior genetic models. *Behavior Genetics*, 16 (1), 163-200.
- McArdle, J.J. (1980). Causal modeling applied to psychonomic systems simulation. *Behavior Research Methods and Instrumentation*, 12, 193-209. (Note: Presented at the 1979 meetings of the *Psychonomic Society*, Phoenix).
- McArdle, J.J. (1979). The development of general multivariate software. In J. Hirschbuhl (Ed.), *Proceedings of the Association for the Development of Computer-Based Instructional Systems*. Akron, Ohio: University of Akron Press. (Note: Presented at the 1979 meetings of the *American Statistical Association*, San Diego).

- McArdle, J.J. (1978). A structural view of structural models. Paper presented at the *Winter Workshop on Latent Structure Models Applied to Developmental Data*, University of Denver, December. (Note: Also presented at the 1979 Meetings of *Psychometric Society*, Monterey, and the *American Psychological Association*, New York.).
- McArdle, J.J. & Aber, M.S. (1990). Patterns of change within latent variable structural equation models. In A. von Eye & M. Rovine (Eds.), *Statistical Methods in Longitudinal Research, Volume I: Principles and Structuring*. New York, Academic Press, 151-224.
- McArdle, J.J. & Anderson, E. (1989). Latent variable growth models for research on aging. In J.E. Birren & K.W. Schaie (Eds.), *Handbook of the Psychology of Aging*. New York: Academic Press, pps. 21-44.
- McArdle, J.J. & Epstein, D.B. (1987). Latent growth curves within development structural equation models. *Child Development*, 58 (1), 110-133.
- McArdle, J.J. & Goldsmith, H.H. (1990). Alternative common factor models for multivariate biometric analyses. *Behavior Genetics*, 30 (5), 583-622.
- McArdle, J.J. & McDonald, R.P. (1984). Some algebraic properties of the Reticular Action Model for moment structures. *British Journal of Mathematical and Statistical Psychology*, 37, 234-251.
- McArdle, J.J. & Prescott, C.A. (1990). Age-based Construct Validation using Structural Equation Modeling. *Experimental Aging Research*, (in press).
- McDonald, R.P. (1987). Describing the elephant: Structure and function in multivariate data. *Psychometrika*, 51 (4), 513-534.
- McDonald, R.P. (1985). *Factor Analysis and Related Methods*. Hillsdale, NJ: Erlbaum.
- McDonald, R.P. (1981). A simple comprehensive model for the analysis of covariance structures: Some remarks on applications. *British Journal of Mathematical and Statistical Psychology*, 33, 161-183.
- McDonald, R.P. (1979). The structural analysis of multivariate data: A sketch of a general theory. *Multivariate Behavioral Research*, 14, 21-28.
- McDonald, R.P. (1978). A simple comprehensive model for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, 31, 59-72.
- McDonald, R.P. & Krane, W.R. (1979). A Monte Carlo study of local identifiability and degrees of freedom in the asymptotic likelihood ratio test. *British Journal of Mathematical and Statistical Psychology*, 32, 121-132.
- Mooijaart, A. (1986). Review of R.P. McDonald's "Factor analysis and related methods". *Psychometrika*, 51 (3), 501-502.
- Mulaik, S. (1986). Factor analysis and Psychometrika: Major developments. *Psychometrika*, 51 (1), 23-34.
- Munck, I.M.E. (1979). *Model Building in Comparative Education: Applications of the LISREL Method to Cross-National Survey Data*. Stockholm: Almqvist & Wiksell.
- Neale, M.C. & McArdle, J.J. (1990). The Analysis of Assortative Mating: A LISREL Model. *Behavior Genetics*, 20 (2) (1), 287-298.
- Nesselroade, J.R. & McArdle, J.J. (1985). Multivariable causal modeling in alcohol use research. *Social Biology*, 32 (3-4), 272-296.
- Rapoport, A. (1972). The uses of mathematical isomorphism in general systems theory. In G. J. Klir (Ed.), *Trends in General Systems Theory*. New York: Wiley.
- Rindskopf, D. (1983). Parameterizing inequality constraints on unique variances in linear structural models. *Psychometrika*, 48, 73-83.
- Short, R., Horn, J.L. & McArdle, J.J. (1984). Mathematical-statistical model building in analysis of developmental data. In R.N. Emde & R.J. Harmon (Eds.), *Continuities and Discontinuities in Development*. New York: Plenum Press, pps. 371-401.
- Sorbom, D. (1974). A general method for studying differences in factor means and factor structure between groups. *Brit. Jour. Math. Statis. Psych.*, 27, 229-239.
- Spearman, C. (1904). General intelligence objectively determined and measured. *American Journal of Psychology*, 15, 201-293.
- Specht, D.A. (1975). On the evaluation of causal models. *Social Science Research*, 4, 113-133.
- Steiger, J.H. (1982). *The general path-analysis moment-structure model of J.J. McArdle*. Department of Psychology, University of British Columbia, Vancouver, B.C.
- Steiger, J.H. (1989). *EzPATH: Causal Modeling. A Supplementary Module for SYSTAT and SYGRAPH*. Evanston, Illinois: SYSTAT, Inc.
- Tufte, E. (1984). *The Graphic Display of Quantitative Information*. Princeton: Graphic Press.
- Van de Geer, J.P. (1971). *Introduction to multivariate analysis for the social sciences*. San Francisco: Freeman.
- Weeks, D.G. (1980). A second-order longitudinal model of ability structure. *Multivariate Behavioral Research*, 15 (3), 353-365.
- Weeks, D.G. (1978). Structural equation systems on latent variables within a second-order measurement model. Unpublished doctoral dissertation, UCLA, August. (Note: Presented at the *Winter Workshop on Latent Structure Models Applied to Developmental Data*, University of Denver, December.).
- Werts, C.E., Joreskog, K.G. & Linn, R.L. (1970) Identification and estimation in path analysis with unmeasured variables. *American Journal of Sociology*, 78 (6), 1469-1484.
- Werts, C.E. & Linn, R.L. (1970) Path analysis: Psychological examples. *Psychological Bulletin*, 74, 193-212.
- Widaman, K.F. (1987). A review of R.P. McDonald's "Factor analysis and related methods." *Jour. Educational Statistics*, 12 (3), 308-313.

- Wiley, D.E. (1973). The identification problem for structural equation models with unmeasured variables. In S. S. Goldberger & O. D. Duncan (Eds.), *Structural Equation Models in the Social Sciences*. New York: Seminar Press.
- Wiley, D.E. Schmidt, W.H. & Bramble, W.J. (1973). Studies of a class of covariance structure models. *Jour. Amer. Stat. Assoc.*, 68, 317-323.
- Wright, S. (1982). On "Path analysis in genetic epidemiology: A critique". *American Journal of Human Genetics*, 35, 757-762.
- Wright, S. (1978). The application of path analysis to etiology. In N.E. Morton & C.S. Chung (Eds.), *Genetic Epidemiology*, New York: Academic Press, 13-54.
- Wright, S. (1934). The method of path coefficients. *Annals of Mathematical Statistics*, 5, 161-215.
- Wright, S. (1921). Correlation and causation. *Journal of Agricultural Research*, 20, 557-585.
- Wright, S. (1918). On the nature of size factors. *Genetics*, 3, 367-374.

INDEX

! D-14
 " D-3
 # D-3
 % D-3
 ; P-4, D-3
 ? D-9, D-14, D-22
 /* P-3, P-5, D-3, D-4
 */ P-4, P-5, D-3, D-4
 () D-3
 0 D-22
 1 D-22
 @ D-9, D-14, D-22
 <Prt Scr> P-4, D-7
 *.FON I-3
 *.OUT P-9, P-18, D-2, D-16, D-17, D-18
 *.RAM I-3, D-2
 *.PS D-2, D-6
 *.SAV D-2, D-19

- A -

A matrix P-11, P-16, M-9, M-10, D-14, D-18
 ACOVSM I-5
 actions M-13, M-14
 address of Data Transforms I-7
 algebraic principles M-8
 aliases P-6, D-3, D-5, D-9, D-22
 arrows I-2, P-5, P-6, P-10, P-15, P-16, D-9, D-13, D-29
 ASCII textfile P-6
 asymmetric parameters P-6
 at sign (@) D-9, D-14, D-22
 axioms M-12, M-13, M-14

- B -

bridges P-11, P-15, M-14

- C -

C matrix P-16, M-9, M-12, D-18
CALIS I-3, I-5, P-20, M-18, M-19, M-22, D-19
CANPATH.RAM P-19
circles P-5, P-6, P-8, D-12, D-25
column D-12
COFAMM I-5
color D-13
 gray D-20
 grayscale D-13
 invisible D-20
 red D-20
 white D-20
 yellow D-20
Command_Name P-4
commands P-4, D-3
comments P-4, P-5, D-3, D-4
compact diagram M-4
component M-14
components of a correlation M-13
connections P-11, P-16, M-9, M-14
constant M-8, D-9
correlation P-8, M-13, M-16
COSAN I-2, I-5, M-2, M-17, M-19, M-24
COSAN-II M-17
covariance P-8
cross-product M-13

- D -

Data Transforms' address I-7
debugging D-24
defaults D-3, D-22, D-28
 default SET_ERROR D-22
 default SET_VALUE D-28
#DEFINE D-30
direct effects M-14, M-17
DRAW commands P-5
 DRAW_BRIDGES P-11, D-6
 DRAW_MODEL P-4, P-9, D-7
DUNCAN.RAM P-20

- E -

E matrix P-11, P-16, M-9, M-12
E-Mail address of Data Transforms I-7
edge P-6, M-13
effects P-11, P-16, M-9, M-14, D-18
effects of length zero M-17
ellipse P-6
empty matrix M-9
EQUAL M-18
equality constraint M-18, D-12
error M-18, D-12
EXAMPLE.OUT I-4
EXAMPLE.PS I-4
EXAMPLE.RAM I-3, I-4
examples P-19, P-20
exclamation point (!) D-14
expectation parameter P-6
expectations P-16, P-19, M-16
EzPath I-5, M-24

- F -

F matrix P-15, P-16, M-9, M-10, D-14, D-18
factor analysis M-2, M-3
factor loading M-3
FACTOR.RAM P-19
FEEDBACK.RAM P-20
filter P-10, P-15, M-10
first command D-27
fixed M-18, D-9, D-22
fixed parameter M-18
free M-18, D-9, D-22
free parameter M-18
.FON I-3
Fontrix I-6
FROMSAS.RAM P-20

- G -

GENERAL commands P-5, P-6
given variable P-6, M-13
graph M-12
GRAPHICS.SYS D-7
Graphtrix I-6
gray color D-20
grayscale D-13

- H -

HAUSER.RAM P-20
hidden variable P-6, M-13
HORN.RAM P-20

- I -

indirect effects M-14, M-17
INPUT commands P-5
 INPUT_CLEAR D-8
 INPUT_MODEL P-4, P-10, D-9
 INPUT_MODEL_MATRIX D-14
input script P-8
intercept M-8
internal model list P-9
invisible D-9, D-12, D-25
 color D-20
 parameter D-13
 shape D-25
 variable D-12

- J -

JORESKOG.RAM P-20

- K -

keywords D-3, D-5, D-9

X - 4

- L -

latent D-9
 growth curve M-7
 variables P-5, P-6, P-8, P-15, M-13
LATENT.RAM P-19
LAWLEY.RAM P-20
length of label D-23
less D-19
level M-7
line M-13
linear structural equation model M-12
link M-13
LISREL I-2, I-3, I-5, M-2, M-17, M-19, M-22
LISREL-7 M-17, M-22, M-23
list structure P-4

- M -

manifest D-9
manifest variables P-5, P-6, P-8, P-15, M-13
matrix D-12
 A matrix P-11, P-16, M-9, M-10, D-14, D-18
 C matrix P-16, M-9, M-12, D-18
 E matrix P-11, P-16, M-9, M-12
 empty matrix M-9
 F matrix P-15, P-16, M-9, M-10, D-14, D-18
 INPUT_MODEL_MATRIX D-14
 OUTPUT_MODEL_MATRIX D-18
 S matrix P-11, P-16, M-9, M-10, D-14, D-18
 sparse matrix M-9
 X matrix P-16, M-12, M-18
maximum length of label D-23
MC interpreter D-3
MCAHORN.RAM P-20
MCARDLE.RAM P-20
MCDONALD.RAM P-20
mean M-8
mediational variables P-7
MIMIC model M-6

X - 5

model:

DRAW_MODEL P-4, P-9, D-7
estimation M-19
expectations M-4
INPUT_MODEL P-4, P-10, D-9
linear structural equation model M-12
list D-9
MIMIC model M-6
multiple regression model M-16
OUTPUT_MODEL P-4, D-17
parameters M-16
RAM M-10
Reticular Action Model M-2
SAVED_MODEL D-19

more D-19

movements M-13

MS-DOS P-2

multiple regression model M-16

- N -

nodes M-13

non-parallel bridges M-14

nonlinear structure M-13

normal D-19

number one (I) D-22

number zero (O) D-22

numeric option D-19

- O -

observed variables P-6, P-15, M-13

one (I) D-22

one-headed arrows P-5, P-8, P-16, M-3, M-10

order of graph M-13

.OUT P-9, P-18, D-2, D-16, D-17, D-18

Output card option M-22

OUTPUT commands P-5

OUTPUT_BRIDGES P-11, D-16

OUTPUT_MODEL P-4, D-17

OUTPUT_MODEL_MATRIX D-18

- P -

Parameter:

Exclusion Rule M-16

expectation P-6

fixed M-18

free M-18

Inclusion Rule M-16

invisible D-13

regression P-6

scaled D-12

parameters P-6, M-13

asymmetric P-6

model M-16

symmetric P-6

parentheses D-3

path analysis M-2, M-4, M-5

path analysis diagrams M-5

PATH.RAM P-19

PATHI M-17

paths M-17

pattern D-12

percent sign (%) D-3

points M-13

postscript P-5, P-9, D-6

pound sign (#) D-3

primitives M-12, M-13, M-14

Print Screen P-4, D-7

Printrix I-6

.PS D-2, D-6

- Q -

question mark (?) D-9, D-14, D-22

quotes (") D-3

- R -

RAM:

acronym M-2
model M-10
notation M-2, M-16
.RAM I-3, D-2
theory I-5, M-8

RAMit I-3

RAMpath:

EXAMPLE I-3
.EXE I-3
prefix D-2

recency D-3

rectangle P-6

red color D-20

REGRESS.RAM P-19

regression:

coefficients P-8, M-8, M-9
effects P-7
model M-16
parameter P-6

Reticular Action Model M-2

- S -

S matrix P-11, P-16, M-9, M-10, D-14, D-18

SAS/CALIS I-3, I-5, P-20, M-18, M-19, M-22, D-19

SAVED_MODEL D-19

.SAV D-2, D-19

scaled parameter D-12

script language P-4

self-slings M-16

semicolon (;) P-4, D-3

SET commands P-5, P-6

SET_COLOR D-20

SET_EQUAL D-21

SET_ERROR D-22

SET_LABEL D-23

SET_PRINT D-19, D-24

SET_SHAPE D-25

SET_SLING_ROTATE D-26

SET_TITLE P-4, D-27

SET_VALUE D-28

SET_WIDTH D-29

shape D-12, D-25

SIMPLEX.RAM P-20

size of graph M-13

slings I-2, P-2, P-5, P-6, P-8, P-10, P-15,
P-16, P-17, M-16, D-9, D-13, D-29

slope M-7

span M-14

sparse matrix M-9

Spearman, Charles M-3

SPEARMAN.RAM P-20

SQRT.RAM P-2, P-19

squares P-5, P-6, P-8, D-12, D-25

standard deviation M-8

standard error M-18, D-12

statistical estimation M-17

symmetric parameters P-6

- T -

text option D-19

tracing M-13, M-14

tracing rules P-5, P-11, M-5, M-8, D-6

triangle M-8, D-12, D-25

TWINA.RAM P-20

two-headed arrows P-5, P-8, P-16, M-3, M-4

- U -

unobserved variables P-6, P-15, M-13, M-16

- V -

Variables: P-5, P-6, P-15, M-13, D-9
disturbance M-16
error M-16
Exclusion Rule M-16
given P-6, M-13
hidden P-6, M-13
Inclusion Rule M-16
invisible D-12
latent P-5, P-6, P-8, P-15, M-13
manifest P-5, P-6, P-8, P-15, M-13
mediational P-7
observed P-6, P-15, M-13
unobserved P-6, P-15, M-13, M-16
variance P-8, M-7, M-8, M-16
vertex P-6, M-13

- W -

white color D-20
WIDAMAN.RAM P-20
width: M-13, M-14, D-13, D-29
of arrows D-29
of bridge M-14
of slings D-29
WRIGHT.RAM P-20

- X -

X matrix P-16, M-12, M-18

- Y -

yellow color D-20

- Z -

zero (0) D-3, D-22
z-level M-8
z-scores M-8
z-slope M-8

X - 10

About the Printing

This manual was typeset entirely with Printrix software from Data Transforms. All of the diagrams were drawn with RAMpath and were then converted from postscript to TIFF files and inserted into the text using Printrix. The fonts used are Caslon50, Press35 and Helvet40 which were created using Fontrix software from Data Transforms.

Johns Hopkins University
Library
370 North Charles Street
Baltimore, MD 21205
(410) 516-7231

About the Authors

J. J. McArdle is a quantitative psychologist. He received a BA from Franklin & Marshall College in 1973, an MA from Hofstra University in 1975, and the Ph.D. from Hofstra University in 1977. McArdle spent six years in a post-doctoral position in psychometrics with John L. Horn of the University of Denver. He moved to the University of Virginia in 1983, and he is currently an Associate Professor of Psychology and Director of the Jefferson Psychometric Laboratory. McArdle has published several papers on the topic of the graphic display of structural models, including original work on RAM theory and the RAMit program. He has also done substantive work on the analysis of growth and change, especially in the area of cognitive aging. In 1987 McArdle was awarded the R. B. Cattell Award for "Distinguished Contributions to Multivariate Research."

Steven M. Boker is a computer software author and data analysis consultant. He received his BS in mathematics from the University of Denver in 1972. He is president of Data Transforms, Inc. a computer software publishing company he founded in 1980. Boker is the author of numerous software programs including Graphtrix, Fontrix and Printrix. His work in graph theory led to the graph expansion algorithm used in RAMpath. Boker is also involved in digital spectral analysis of medical data, neural network algorithms and scientific visualization. He is currently doing research on the applications of chaos modeling to behavioral aging research.

**Naval Health Research Center
Wilkins Biomedical Library
140 Sylvester Road
San Diego, CA 92106-3521**

RAMpath

Path Diagram Software

